# BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

## IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.**

(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Offenlegungsschrift**

(10) **DE 198 10 784 A 1**

(51) Int. Cl.[6]:
**G 06 F 9/445**
G 06 F 11/28

(21) Aktenzeichen: 198 10 784.6
(22) Anmeldetag: 12. 3. 98
(43) Offenlegungstag: 16. 9. 99

(71) Anmelder:
Telefonaktiebolaget L M Ericsson (publ), Stockholm, SE

(74) Vertreter:
HOFFMANN · EITLE, 81925 München

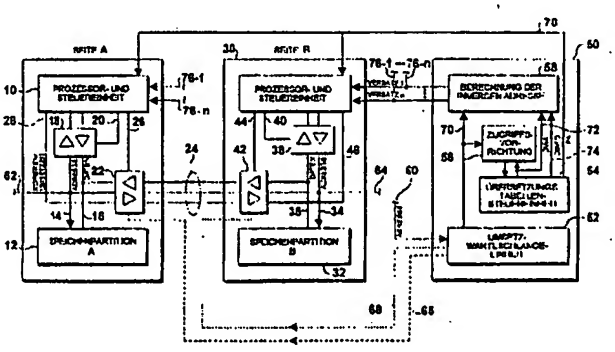
(72) Erfinder:
Gard, Bengt Erik Ingemar, Tullinge, SE; Kling, Lars-Örjan, Södertälje, SE; Johnsson, Sten Edvard, Bandhagen, SE

**Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen**

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Hardwareunterstützung für Datenumsetzung

(57) Zum Erzielen einer hochwirksamen Aktualisierung von Software in computerbasierten Systemen wird ein Software-Bearbeitungssystem mit zwei Partitionen (A, B) geschaffen, die über eine Verbindungsvorrichtung (24) verbunden sind. Die beiden Partitionen (A, B) aktualisieren einen Zustand der neuen Software in einer Speichervorrichtung (12) gemäß dem Zustand einer alten Software in der anderen Speichervorrichtung (32) während der Ausführung der alten Software. Insbesondere wird die Übertragung von Daten von der alten Software zu der neuen Software durch eine Datenumsetzungs-Unterstützungsvorrichtung (50) unterstützt, die die Startadresse eines Umsetzprogramms im Zusammenhang mit der umzusetzenden Datenvariablen ausgibt.



DE 198 10 784 A 1

DE 198 10 784 A 1

Beschreibung

## GEBIET DER ERFINDUNG

Die vorliegende Erfindung betrifft eine Hardwareunterstützung bei einer Datenumsetzung für den Einsatz in einem Zustandskopierverfahren bei der Aktualisierung von Software.

## HINTERGRUND DER ERFINDUNG

Beim Durchführen der Aktualisierung von Software tritt üblicherweise eine Störung des Betriebs bei dem zu aktualisierenden System auf. Diese Störung reicht von einer vollständigen Systemunterbrechung während mehrerer Stunden oder möglicherweise mehrerer Tage bis zu einer kurzen Unterbrechung bei möglicherweise lediglich einem begrenzten Teil der gesamten Systemfunktionalität, beispielsweise einige Sekunden. Es kann auch keine Störung wahrnehmbar sein, obgleich dies typischerweise bei tatsächlich vorliegenden Systemen nicht der Fall ist. Jedoch ist es für Systeme wie Kommunikationsvermittlungen von größter Wichtigkeit, daß jede Störung so gering und so kurz wie möglich ist.

Zum Erzielen einer störungsfreien Aktualisierung von Software selbst mit Softwaremodulen, die fortlaufend lang betrieben werden, besteht die Anforderung, daß die neue Software mit den gesamten erforderlichen Daten der alten Software aktualisiert wird, während die alte Software fortlaufend ausgeführt wird. Haben Daten der neuen Software denselben Zustand wie die Daten der alten Software erreicht, so übernimmt die neue Software den Betrieb.

Das einfachste Verfahren zum Aktualisieren von Software besteht in der Unterbrechung der Ausführung der alten Software, dem Laden der neuen Software und schließlich dem Start der neuen Software. Bei Einsatz dieses Verfahrens werden keine Daten zwischen der alten Software und der neuen Software übertragen. Ferner gehen sämtliche aktivierte Softwareprozesse verloren, und die Ausbildung von Software wird während dem Laden und dem Start der neuen Software gestoppt. Üblicherweise wird dieses Verfahren beispielsweise für Workstations und Personal-Computer eingesetzt.

Eine verbesserte Vorgehensweise für ein Kommunikationssystem wurde in "Fernhochstufung und Aktualisierung von AXE 10 Software", Seite 66, 67 Ericsson Review Nr. 2, 1996 beschrieben. Hier wird die neue Software geladen, während die alte Software weiter das Bereitstellen von Diensten für Teilnehmer handhabt. Daten werden zwischen der alten Software und der neuen Software übertragen. Datenvariablen mit zu übertragenden Daten werden in der sogenannten Datenveränderungsinformation identifiziert, und sie können entweder vom Typ Kopieren oder Umsetzen sein. Für jede umzusetzende Datenvariable ist in der Datenveränderungsinformation ein Umsetzprogramm spezifiziert, das zum Erzielen der Transformationen und zum Übertragen des Umsetzergebnisses an die neue Software ausgeführt wird. Jedoch wird während der Übertragung von Daten für bereits eingerichtete Dienste von der alten Software zu der neuen Software das Bereitstellen zusätzlicher Dienste unterbrochen.

Eine die genannten Vorgehensweisen unterstützende Hardwarestruktur ist in **Fig. 16** gezeigt. Damit die alte Software fortlaufend ausgeführt werden kann, während die neue Software geladen wird, ist die Systemarchitektur in eine Seite A und eine Seite B partitioniert. Die Seite A enthält eine erste Prozessor- und Steuereinheit **10**, sowie eine erste Speicherpartition **12**. Die Speicherpartition speichert Daten und Softwaremodule, die in der Prozessor- und Steuereinheit **10** ausgeführt und durch diese gesteuert werden. Für den Austausch von Daten und Softwareinformation ist ein Adreßbus **14** und ein Datenbus **16** vorgesehen, der durch eine Datenadreßbus-Steuereinheit **18** gesteuert wird. Diese Datenadreßbus-Steuereinheit **18** ist mit der Prozessor- und Steuereinheit **10** über eine Datenadreßbus-Steuerleitung **20** verbunden.

Ferner ist eine Aktualisierungsbus-Steuereinheit **22** vorgesehen, die den Adreßbus **14** und den Datenbus **16** mit einem externen Aktualisierungsbus **24** verbindet der für die Aktualisierung der Software vorgesehen ist. Die Aktualisierungsbus-Steuereinheit **22** ist mit der Prozessor- und Steuereinheit **10** über eine Aktualisierungsbus-Steuerleitung **26** für die Steuerung verbunden. Weiterhin besteht eine Option im Weiterleiten, Kopieren/Umsetzen von Information im Zusammenhang mit zu übertragenden Daten über den Aktualisierungsbus **24** an die Aktualisierungsbus-Steuereinheit **22** über eine Kopier/Umsetz-Datenleitung **28**.

Wie in **Fig. 16** gezeigt, gilt dieselbe Struktur, wie sie im Zusammenhang mit der Seite A erläutert wurde, auch für die Seite B. Deshalb ist bei der Seite B eine zweite Prozessor- und Steuereinheit **30** und eine zweite Speicherpartition **32** vorgesehen. Die zweite Prozessor- und Steuereinheit **30** und die zweite Speicherpartition **32** sind durch einen zweiten Adreßbus **34** und einen zweiten Datenbus **36** angeschlossen, die beide durch eine zweite Datenadreßbus-Steuereinheit **38** gesteuert werden. Diese zweite Datenbus-Steuereinheit **38** ist mit der zweiten Prozessor- und Steuereinheit **30** über eine zweite Daten-Adreßbus-Steuerleitung **40** verbunden. Zum Erzielen des externen Austausches von Daten, ist eine zweite Aktualisierungsbus-Steuereinheit **42** vorgesehen, die mit der zweiten Prozessor- und Steuereinheit **30** über eine zweite Aktualisierungsbus-Steuerleitung **44** verbunden ist. Zu kopierende/umzusetzende Daten, die jeweils zu und von der Seite A/B zu übertragen sind, werden an die zweite Aktualisierungsbus-Steuereinheit **42** über eine zweite Kopier/Umsetz-Datenleitung **46** übertragen.

Wie in **Fig. 16** gezeigt, ermöglicht diese bipartitionierte Architektur das fortlaufende Ausführen alter Software, während die neue Software geladen wird und ein Kopieren und Umsetzen von Datenvariablen mit zu übertragenden Daten entweder von der Seite A zu der Seite B oder umgekehrt erfolgt.

Jedoch besteht ein Problem bei dieser Vorgehensweise darin, daß sich die gesamten Systemeigenschaften während der Aktualisierung der Software verschlechtern. Insbesondere werden üblicherweise während der Übertragung von Daten die Dienste gestoppt, um eine konsistente Kopie der Daten der alten Software und der neuen Software zu erhalten.

Ein weiterer Nachteil im Hinblick auf die in **Fig. 16** gezeigte bipartitionierte Architektur besteht darin, daß keine spezifische Hardwareunterstützung für die Datenumsetzung vorgesehen ist. Demnach kann die softwarebasierte Umsetzung von Daten zeitaufwendig sein und zu erhöhten Systemunterbrechungszeiten führen, was wiederum einen Verlust an Einnahmen für den Dienstanbieter bedeutet, der diese Hardwarestruktur einsetzt.

Zum Überwinden dieses Nachteils wurde in US-A-5 155 837 vorgeschlagen, die Eingabe von Daten für neue Dienste zu der neuen Software in einem ersten Schritt umzuschalten. Sind die auf der Grundlage der alten Software ablaufenden Dienste sämtlich beendet, so wird die Ausgabe für Daten dieser fortschreitenden Dienste anschließend von der alten Software zu der neuen Software umgeschaltet. Demnach müssen Dienste im Zusammenhang mit der alten Software abgeschlossen sein, bevor die neue Software betriebsbereit ist, und somit ist es lediglich möglich, Dienste mit einer sehr

kurzen Dauer handzuhaben. Weiterhin ist in US-A-555 837 eine Hardwareunterstützung für eine Datenumsetzung nicht offenbart.

## ZUSAMMENFASSUNG DER ERFINDUNG

Im Hinblick auf die obigen Ausführungen besteht die Aufgabe der Erfindung in der Schaffung einer hochwirksamen Aktualisierung von Software in computerbasierten Systemen.

Im Rahmen der vorliegenden Erfindung wird diese Aufgabe gelöst durch ein Softwarebearbeitungssystem vom partitionierten Typ, enthaltend eine erste Partition mit einer ersten Speichervorrichtung, eine zweite Partition mit einer zweiten Speichervorrichtung, derart, daß die erste Partition und die zweite Partition mit einer Verbindungsvorrichtung verbunden sind und einen Zustand einer neuen Software in einer Speichervorrichtung an den Zustand einer alten Software in der anderen Speichervorrichtung während der Ausführung der alten Software anpassen, und zum Unterstützen der Übertragung von Daten von der alten Software zu der neuen Software eine Datenumsetz-Unterstützungsvorrichtung für die Ausgabe der Startadresse eines Umsetzprogramms im Zusammenhang mit umzusetzenden Datenvariablen vorgesehen ist.

Somit wird im Rahmen der vorliegenden Erfindung eine Hardwarelösung zum Erzielen der erforderlichen Übertragung und Umsetzung von Daten im Zusammenhang mit der alten Software und der neuen Software während der Aktualisierung von Software in computerbasierten Systemen geschaffen, beispielsweise eine Aktualisierung aufgrund hinzugefügter Funktionalität und/oder aufgrund der Korrektur von Fehlern.

Ferner wird gemäß der vorliegenden Erfindung eine an eine Übersetzungsspeichervorrichtung angepaßte Zugriffsvorrichtung zum Bestimmen der Speicheradresse vorgegebener Datenwerte in einem Speichergerät geschaffen, derart, daß die optimalen Datenwerte (gemäß einem Sonderfall) in sortierter Folge bei Speicheradressen gemäß einer binären Baumdatenstruktur von Knoten, Zweigen, Unterbäumen und Blättern gespeichert sind, und die Zugriffsvorrichtung eine Auslesevorrichtung zum Auslesen eines Datenwerts bei einer momentanen Suchadresse aus dem Speichergerät enthält, sowie eine Vergleichsvorrichtung zum Vergleichen des ausgelesenen Datenwerts mit dem zu suchenden Datenwert für die Bestimmung der Tatsache, ob der ausgelesene Datenwert größer oder kleiner als der zu suchende Datenwert ist, und eine Bestimmungsvorrichtung zum Bestimmen einer vollständigen nächsten Suchadresse, bei der nach dem Datenwert zu suchen ist, und zwar auf Basis des Vergleichsergebnisses und der momentanen Suchadresse, derart, daß die Auslesevorrichtung, die Vergleichsvorrichtung und die Bestimmungsvorrichtung das Auslesen, das Vergleichen und das Bestimmen rekursiv solange durchführen, bis der ausgelesene Datenwert mit dem zu suchenden Datenwert bis auf eine vorgegebene Toleranz übereinstimmt.

Somit unterscheidet sich die Zugriffsvorrichtung gemäß dem Sonderfall signifikant von üblichen Verfahren, bei denen ein speicherintensives Zeigerschema zum Bestimmen der Adresse eines Knotens in dem binären Baum eingesetzt wird. Insbesondere verändert sich während einer Traversierung des binären Baums der signifikanteste Teil der Adresse bei jedem Schritt während der Traversierung des binären Baums.

Bei einem DRAM-Speicher wäre dies zeitaufwendig mit einem Zeilen/Spalten-Adressierungsschema, bei dem die Zeile im signifikantesten und die Spalte dem am wenigsten signifikanten Teil der Adresse entspricht.

Im Gegensatz hierzu wird im Rahmen der vorliegenden Erfindung eine Zugriffsvorrichtung und ein zugeordnetes Verfahren zum Bestimmen der Speicheradresse eines vorgegebenen Datenwertes in einer Speichereinrichtung ohne Einsatz von Zeigern zum Bestimmen der Speicherstellen während der Suche in binären Baum geschaffen. Weiterhin kann die Zugriffsvorrichtung die Speicheradresse schneller als bekannte Verfahren bestimmen.

Im Rahmen der vorliegenden Erfindung wurde festgestellt, daß keine Anforderung zum Durchführen der häufigen Veränderungen des signifikantesten Teils der Adresse besteht und daß sowohl der Spaltenteil der Adresse als auch der Zeilenteil der Adresse auf der Basis des Vergleichsergebnisses und der momentanen Adresse zu bestimmen sind. Bei der vorliegenden Erfindung ermöglicht die Kombination des Spaltenadreßteils der Abbildung und des Zeilenadreßteils der Abbildung die Verringerung der Suchzeit, bis ein Datenwert in dem Speicher lokalisiert wird.

Gemäß der vorliegenden Erfindung wird dann, wenn ein Blatt eines Unterbaums während der Suche erreicht wird, ein neuer Wurzelknoten eines anderen Unterbaums bestimmt. Wird in diesem Unterbaum der Datenwert immer noch nicht gefunden, so geht die Suchprozedur von dem Blattknoten des momentanen Unterbaums zu einem weiteren Wurzelknoten eines anderen Unterbaums, bis ein unterster Blattknoten des vollständigen Baums erreicht wird. Da eine derartige Suche immer eine Traversierung des Baums von der Wurzel bis zu dem Blattknoten entspricht, führt die Suche hauptsächlich zu Traversierungen innerhalb der Unterbäume. Nur eine minimale Zahl der Schritte führt zu einer Veränderung der Unterbäume. Als Ergebnis der Abbildung der vorliegenden Erfindung verändert sich lediglich die Spaltenadresse während einer Traversierung innerhalb eines Unterbaums. Demnach ist die Zahl der Veränderungen der Zeilenadresse minimiert, d. h. die Veränderungen der Zeilenadresse werden minimal gehalten. Deshalb wird die Zeit für eine Suche erheblich reduziert.

## KURZE BESCHREIBUNG DER ZEICHNUNG

Bevorzugte Ausführungsformen der vorliegenden Erfindung werden unter Bezug auf die beiliegende Zeichnung beschrieben; es zeigen:

**Fig. 1** die Architektur der in Hardware realisierten Unterstützungseinrichtung für die Datenumsetzung gemäß der vorliegenden Erfindung;

**Fig. 2** unterschiedliche Schritte des Zustandskopierverfahrens, das von der hardwarebasierten Unterstützung für die Datenumsetzung gemäß der vorliegenden Erfindung umgesetzt wird;

**Fig. 3** ein Flußdiagramm gemäß dem in **Fig. 2** dargestellten Zustandskopierverfahren;

**Fig. 4** die Architektur der in **Fig. 1** gezeigten Übersetzungstabellen-Speichereinheit;

**Fig. 5** eine Speichereinrichtung der in **Fig. 1** gezeigten Zugriffsvorrichtung, bei der Datenwerte wahlfrei bei spezifischen Adreßpositionen gespeichert sind;

**Fig. 6** eine logische Binärbaumstruktur zum Durchführen einer Binärbaumsuche in der Speichereinrichtung der in **Fig. 1** gezeigten Zugriffsvorrichtung;

**Fig. 7** eine Binärbaumstruktur unter Einsatz von Unterbäumen gemäß der vorliegenden Erfindung;

**Fig. 8** eine Speichereinrichtung, bei der jede Zeile auf einen der in **Fig. 7** gezeigten Unterbäume abgebildet ist;

**Fig. 9** die Organisation der Daten in dem in **Fig. 8** gezeigten Unterbaum;

**Fig. 10** ein Flußdiagramm zum Darstellen des Suchverfahrens, das von der Zugriffseinheit für den Spezialfall des

Einsatzes einer Binärbaumstruktur gemäß der vorliegenden Erfindung eingesetzt wird;

**Fig.** 11 ein Beispiel für in einem Speicher gemäß einer Binärbaumstruktur gespeicherten Datenwerte;

**Fig.** 12 den Binärbaum und zugeordnete Positionen der Knoten gemäß dem in **Fig.** 11 gezeigten Beispiel;

**Fig.** 13 eine Ausführungsform der Zugriffseinheit gemäß der vorliegenden Erfindung für den Zugriff auf Datenwerte, die in der in **Fig.** 1 gezeigten Übersetzungstabellen-Speichereinheit gespeichert sind;

**Fig.** 14 ein Beispiel für die inverse Adressenberechnung, die von der in **Fig.** 1 gezeigten inversen Adressen-Berechnungseinheit durchgeführt wird;

**Fig.** 15 ein anderes Beispiel für die inverse Adreßberechnung, die von der in **Fig.** 1 gezeigten Inversadressen-Berechnungseinheit durchgeführt wird;

**Fig.** 16 eine Architektur für ein Softwarebearbeitungssystem vom partitionierten Typ gemäß dem technologischen Hintergrund der Erfindung; und

**Fig.** 17 Zusätze zu dem in **Fig.** 10 dargestellten Algorithmus zum Festlegen von Abschlußbedingungen gemäß Gl. **(13)**.

## BESCHREIBUNG BEVORZUGTER AUSFÜHRUNGSFORMEN

Die **Fig.** 1 zeigt die Grundarchitektur der Hardwareunterstützung für die Datenumsetzung gemäß der vorliegenden Erfindung. Teile, die mit den zuvor unter Bezug auf die **Fig.** 16 gezeigten übereinstimmen oder diesen ähnlich sind, sind mit demselben Bezugszeichen bezeichnet, und ihre Beschreibung wird hier nicht wiederholt.

Wie in **Fig.** 1 gezeigt, unterscheidet sich die Architektur der Hardwareunterstützung für die Datenumsetzung gegenüber bekannten Vorgehensweisen dahingehend, daß eine eigene Umsetzunterstützungseinheit **50** vorgesehen ist. Die Umsetzunterstützungseinheit **50** kann eine eigene Einheit oder eine duplizierte Einheit gemäß den unterschiedlichen Partitionen A, B des Softwarebearbeitungssystems sein, d. h. jeweils für die Seite A und B vorgesehen sein. Die Umsetzunterstützungseinheit **50** enthält eine Umsetz-Warteschlangeneinheit **52**, eine Übersetzungstabelle-Speichereinheit **54**, eine Zugriffseinheit **56** und eine Inversadressen-Berechnungseinheit **58**.

Wie in **Fig.** 1 gezeigt, werden die Adressen der umzusetzenden Datenvariablen der Umsetzunterstützungseinheit **50** über eine Adresseneingabeleitung **60** zugeführt, die sowohl mit der Seite A bei einem Punkt **62** als auch mit der Seite B bei der Stelle **64** verbunden ist.

Die der Umsetzunterstützungseinheit **50** zugeführte Adresse wird in der Umsetz-Warteschlangeneinheit **52** gespeichert. Die Umsetz-Warteschlangeneinheit **52** puffert übertragene Adressen der Datenvariablen und kann, z. B. mit einem dynamischen RAM-Speicher implementiert sein. Besteht zudem eine Anforderung zum zeitweisen Stoppen des Speicherns neuer Daten in der Warteschlange in dem Fall, in dem die Kapazität der Umsetz-Warteschlangeneinheit erreicht ist, so wird ein Rückhaltesignal von der Umsetz-Warteschlangeneinheit **52** entweder zu der ersten Aktualisierungsbus-Steuereinheit **22** bei der Seite A über eine erste Rückhaltesignalleitung **66** oder zu der zweiten Aktualisierungssteuer-Baueinheit **42** über eine zweite Rückhaltesignalleitung **68** gesendet.

Wie in **Fig.** 1 gezeigt, enthält die Umsetzunterstützungseinheit **50** ferner die Übersetzungstabellen-Speichereinheit **54**, die einen Eintrag für jede Adresse aufweist, die gegebenenfalls zu der Umsetz-Warteschlangeneinheit **52** übertragen wird. Der Eintrag ermöglicht die Bestimmung von Information im Hinblick auf die Datenstruktur und die Umsetzsoftware im Zusammenhang mit der Adresse in der Umsetz-Warteschlangeneinheit **52**. Deshalb werden nach dem Zuführen der Adresse zu der Übersetzungstabellen-Speichereinheit **54** als Ausgangsgröße aus der Übersetzungstabelle gemäß der Adresse folgende Werte ausgegeben:

DWA: Startadresse der Datenstruktur im Zusammenhang mit der der Übersetzungstabellen-Speichereinheit **54** zugeführten Adresse;

DINFO: Strukturinformation über die Variable für die Inversadreßberechnung; und/oder Umsetzung; und

PA: Adresse des Umsetzprogramms für die Variable.

Demnach führt die Umsetzunterstützungseinheit als ersten Schritt die Suche nach einem Eintrag in der Übersetzungstabellen-Speichereinheit **54** gemäß einer der Umsetz-Warteschlangeneinheit **52** entnommenen Adresse durch. Für die wirkungsvolle Durchführung der Suche in der Übersetzungstabellen-Speichereinheit **54** ist eine Zugriffseinheit **56** vorgesehen, in der die Suche in der Übersetzungstabelle als Suche für den höchsten Wert DWA durchgeführt wird, der kleiner oder gleich der vorgegebenen und von der Umsetz-Warteschlangeneinheit **52** über eine Adreßzuführleitung **60** zugeführten Adresse ist.

Insbesondere implementiert gemäß einem Spezialfall eine Zugriffseinheit **56** die Suche unter Einsatz eines Binärbaums, wie im folgenden detaillierter unter Bezug auf die **Fig.** 5 bis 12 erläutert wird.

Sobald die Adresse der umzusetzenden Datenvariablen und im Zusammenhang mit der entsprechenden Datenstruktur verfügbar ist, wird diese der Inversadressen-Berechnungseinheit **58** über eine Adreßzuführleitung **70**, eine DWA-Zuführleitung **72** und eine DINFO-Zuführleitung **74** zugeführt. Unter Einsatz dieser Information leitet die Inversadressen-Berechnungseinheit **58** zusätzliche Information im Zusammenhang mit der umzusetzenden Variablen ab, z. B. den Index in einem Feld oder die Indizes für eine Matrix.

Allgemein besteht das Ergebnis aus N Indizes, die entweder der Seite A oder der Seite B jeweils über Versatzzuführleitungen **76-1**, . . ., **76-N** zugeführt werden. Weiterhin wird die Adresse des Umsetzprogramms zum tatsächlichen Durchführen der Umsetzung entweder in der ersten Prozessor- und Steuereinheit **10** oder der zweiten Prozessor- und Steuereinheit **30** hierzu über die in **Fig.** 1 gezeigte Umsetzprogramm-Adreßleitung **78** zugeführt.

Wie in **Fig.** 1 gezeigt, nützt die Hardwareunterstützung für die Datenumsetzung den Aktualisierungsbus **24**. Die Adresse wird zwischen der Seite A und der Seite B oder umgekehrt übertragen. Die Adresse wird jeweils zum Einschreiben in die zu aktualisierende und neu geladene Speicherpartition **32** oder **12** eingesetzt. Bei jeder Seite ist eine (nicht gezeigte) Arbitrierungslogik für den Speicherpartitionszugriff vorgesehen, da unabhängige Anwender der Speicherpartition bei der zu aktualisierenden Seite einen Zugriff ausüben.

Wird ein verzögerter Start der Umsetzprogramme gewünscht, so kann gemäß der vorliegenden Erfindung die Umsetzunterstützungseinheit **50** auch eine (nicht gezeigte) zugeordnete Puffereinheit enthalten, die zum Speichern des Ergebnisses der durch die Zugriffseinheit **56** und die Inversadressen-Berechnungseinheit **58** erzielten Ergebnisses hinzugefügt ist. In diesem Fall verzögert die zugeordnete Puffereinheit der tatsächlichen Start des Umsetzprogramms. Speichert die zugeordnete Puffereinheit niemals denselben Eintrag zweifach, so ist garantiert, daß die gleiche Umsetzung nicht mehrfach während einem vorgegebenen Verzögerungsintervall gestartet wird.

Mit Hilfe der oben beschreibenen Hardwarestruktur wird

der grundlegende Prozeß für die Umsetzung von Daten durchgeführt, wie unter Bezug auf die **Fig. 2** und 3 erläutert. Ohne Beschränkung des Umfangs der Erfindung für die Erklärung sei angenommen, daß Daten von der Seite A zu der Seite B übertragen werden, und daß die neue Software bei der Speicherpartition der Seite B geladen wird.

Wie in **Fig. 2** gezeigt, führen in einem ersten Schritt **1** beide Seiten A und B einen parallelen synchronen Modus unter Ausführung derselben Software durch. Ferner betrifft der in **Fig. 2** gezeigte Schritt **2** das Laden der neuen Software in die Seite B, während die Ausführungsform der alten Software durch die erste Prozessor- und Steuereinheit **10** fortgeführt wird. Im Schritt **3** führt die erste Prozessor- und Steuereinheit **10** das Kopieren von Daten von der Seite A zu der Seite B durch.

Wie in dem unteren Teil im Zusammenhang mit dem Schritt **3** gezeigt, können Daten nicht nur kopiert werden, sondern im Rahmen der vorliegenden Erfindung kann auch ein Umsetzen während der Übertragung zu der Seite B in der Umsetzunterstützungseinheit **50** erfolgen. Hierbei wird die Umsetzung parallel zu und ohne Störung der Ausführungsform der alten Software in der ersten Prozessor- und Steuereinheit **10** durchgeführt, damit eine erhebliche Beschleunigung erzielt wird.

Wie in **Fig. 2** gezeigt, führt im Schritt **4** die zweite Prozessor- und Steuereinheit **30** eine Initialisierung der zweiten Speicherpartition **32** parallel zu und ohne Störung der in der ersten Prozessor- und Steuereinheit **10** ablaufenden alten Software aus. Hierbei wird der Initialisierungsschritt entweder unmittelbar nach dem Laden der neuen Software in der zweiten Speicherpartition **32** gemäß dem Schritt **2** durchgeführt, oder sobald als möglich, sofern sie von Daten abhängt, die von der alten Software im Schritt **3** kopiert werden.

Ferner können Daten im Zusammenhang mit der alten Software lediglich teilweise übertragen werden, und spezielle Initialisierungsschritte werden vor oder unmittelbar nach dem Umschalten zu der Seite B durchgeführt, damit Vorgabeinitialisierungsschritte durchgeführt werden, die keine vollständige Eingabe der Daten der alten Software erfordern.

Wie in **Fig. 2** gezeigt, wird unmittelbar bei Erreichen eines geeigneten Zustands der Seite B in Schritt **5** ein Umschalten zu der Ausführen der neuen Software durch die zweite Prozessor- und Steuereinheit **30** durchgeführt. Hier kann das Umschalten gemäß den einzelnen Softwaremoduln durchgeführt werden, unmittelbar nachdem derselbe Zustand für die jeweiligen Softwaremoduln in beiden Speicherpartitionen **12** und **32** erreicht ist. Liegen Daten im Zusammenhang mit der alten Software vor, die im Zeitpunkt des Umschaltens aufgrund lediglich einer Teilübertragung von Daten nicht übertragen sind, so können, falls erforderlich, diese Daten immer noch vor dem Start der neuen Software übertragen werden.

Wie in **Fig. 2** im Zusammenhang mit den Schritten **3** und **4** zudem gezeigt, wird die Umsetzunterstützungseinheit **50** gemäß der vorliegenden Erfindung fortlaufend auch während dem Initialisierungsschritt für die Speicherpartition der Seite B betrieben. Der Grund hierfür besteht darin, daß Software fortlaufend während dem Aktualisierungsprozeß ausgeführt wird, und bereits zu der Umsetzunterstützungseinheit **50** übertragene Daten überschreiben kann. Demnach werden einer oder mehrere Aktualisierungs- und Umsetzprozesse wiederholt im Hintergrund solange durchgeführt, bis ein Umschalten zu der neuen Software erfolgt, damit der sich verändernde Zustand in der ersten Speicherpartition **10** berücksichtigt wird. Der wiederholte Prozeß kann durch die Umsetzunterstützungseinheit **50** parallel zum Initialisierungsschritt für die Speicherpartition **32** der Seite B durchgeführt werden.

Die **Fig. 3** zeigt ein Flußdiagramm gemäß dem mit Bezug auf die **Fig. 2** erläuterten Aktualisierungsprozeß. Insbesondere ist zu erkennen, daß im Zusammenhang mit einem Vorbereitungsschritt die neue Software geladen wird. Gemäß der vorliegenden Erfindung kann die Umsetzunterstützungseinheit **50** simultan und parallel solange betrieben werden, bis das Umschalten zu der Seite B erfolgt.

Demnach wird gemäß der vorliegenden Erfindung die Ausführungsform der alten Software nicht unterbrochen, wenn die Aktualisierung der neuen Software stattfindet. Die Übertragung von Daten im Zusammenhang mit der alten Software wird mit Hardwareunterstützung durch die Umsetzunterstützungseinheit **50** durchgeführt.

Während vorangehend unter Bezug auf die **Fig. 1** bis 3 die Grundlagen der Hardwareunterstützung für die Datenumsetzung beschrieben wurden, folgt nun unter Bezug auf die **Fig. 4** bis 15 die weitergehende Erläuterung von Details insbesondere im Zusammenhang mit der Übersetzungstabellen-Speichereinheit **54**, der Zugriffseinheit **56** und der Inversadressen-Berechnungseinheit **58**.

Die **Fig. 4** zeigt die interne Struktur der Übersetzungstabellen-Speichereinheit **54**. Wie oben dargelegt, liegt für jede gegebenenfalls durch die Umsetzunterstützungseinheit **50** umzusetzende Datenvariable ein Eintrag $v_i$ aus einer Gruppe $V_s$ aller umzusetzenden Datenvariablen vor:

$$V_s = \{v_1, \ldots, v_i, \ldots, v_s\} \quad (11).$$

Jede Adresse und demnach jeder Eintrag der Übersetzungstabelle betrifft eine Datenstruktur $d_j$:

$$d_j = \{n_j; DWA, \ldots, \alpha_i, \ldots, DINFO, \ldots; \ldots, P_i, \ldots\} \quad (12).$$

Hierbei repräsentiert $n_j$ die Zahl der Variablen in der Datenstruktur, $\alpha_i$ repräsentiert die Adresse der Datenvariablen in der Datenstruktur $d_j$, DINFO repräsentiert Strukturinformation im Zusammenhang mit der Variablen in der Datenstruktur $d_j$ und $P_i$ repräsentiert die Startadresse des der jeweiligen Datenvariablen zugeordneten Umsetzprogramms.

Wie in **Fig. 4** gezeigt, speichert eine Übersetzungstabelle lediglich die Adresse DWA der ersten Variablen der Datenstruktur $d_j$ bei jedem Eintrag $v_i$, sowie die zu suchende Adresse. In anderen Worten ausgedrückt, wird für die unterschiedlichen Datenvariablen einer Datenstruktur $d_j$ immer dieselbe Startadresse DWA durch die Übersetzungstabellen-Speichereinheit **54** ausgegeben.

Ferner betrifft der Eintrag DINFO die Strukturinformation für die zu suchende Datenvariable, z. B. Information dahingehend, ob sie ein eindimensionales Feld, ein zweidimensionales Feld oder eine Matrix oder irgendeine andere flexible Datenstruktur betrifft.

Der dritte Punkt jedes Eintrags in der Übersetzungstabelle ist die Startadresse des der umzusetzenden Datenvariablen zugeordneten Umsetzprogramms entweder bei der Prozessor-Steuereinheit **10** oder **30**.

Für jede der Umsetzunterstützungseinheit **50** zugeführte Adresse $\alpha$ einer Datenstruktur muß der Eintrag $v_i$ in der Übersetzungstabelle gefunden werden, der folgenden DWA-Wert aufweist:

$$v_i = \{v \in V_s \mid DWA_i \leq \alpha \wedge \alpha < DWA_{i+1}\} \quad (13).$$

Die **Fig. 5** bis 13 zeigen eine Ausführungsform der Zugriffseinheit **56** und der Übersetzungstabellen-Speichereinheit

heit **54**, die besonders zum Ausführen dieser Suche in der wirksamsten Weise ausgebildet ist.

Allgemein besteht oft die Anforderung zum Durchführen einer schnellen Suche insbesondere nach einem Datenwert im Zusammenhang mit zugeordneter Information in einem Speicher, in dem eine große Zahl von Einträgen abgelegt sind. Eine weitere Möglichkeit besteht darin, die Speichereinrichtung mit der Vielzahl von Datenwerten sequentiell nach einem bestimmten Datenwert zu durchsuchen und dann, wenn der Datenwert gefunden ist, die zugeordnete Information auszulesen. Sind Datenwerte wahlfrei in der Speichereinrichtung gespeichert, so kann jedoch zum Auffinden des Datenwerts der gesamte Speicher zu durchsuchen sein.

Zum Vermeiden eines vollständigen Durchsuchens der Speichereinrichtung können die Datenwerte auch in der Speichereinrichtung gemäß vorbestimmter Regeln gespeichert sein, z. B. sortiert, was später zum Auffinden des Datenwerts während der Suchprozedur ausgenützt werden kann.

Die **Fig. 5** zeigt ein Beispiel eines zweidimensionalen Speichers, in dem Dateneinträge durch eine Adresse mit einem Spaltenadressenteil A und einem Zeilenadressenteil B, d. h. der Adresse S = < B(X), A(X) >, adressiert werden. X kann als der Knoten in einem Binärbaum angesehen werden, an dem sich die spezifische Zeilenadresse und Spaltenadresse überschneiden. Wie sich der **Fig. 5** entnehmen läßt, können die einzelnen Datenwerte D wahlfrei in der Speichereinrichtung gespeichert werden. Eine andere Möglichkeit würde darin bestehen, eine Abbildung von Datenwerten auf Speicherstellen gemäß einer Binärbaum-Datenstruktur zum Darstellen der sortierten Datenwerte zu verwenden. Beispielsweise ist aus der WO 96/09580 die Verwendung eines Binärbaums bekannt, um eine Sortierung von Aufzeichnungen vorzunehmen.

Wie in **Fig. 6** gezeigt, ist in der Binärbaum-Datenstruktur jeder Knoten (der Schnittpunkt X, an dem ein Datenwert eingegeben werden kann und davon gelesen werden kann) mit zwei anderen Knoten verbunden. Die Suche durch den Binärbaum (oder die Speichereinrichtung, die darauf abgebildete Einträge aufweist) ist wie folgt. Ein Datenwert D1 wird aus einem Knoten $X_1$ (der sich an einer Zeilenadresse B(X) und einer Spaltenadresse A(X) befindet) ausgelesen, und der Datenwert D1 wird mit dem Datenwert I verglichen, dessen Speicheradresse bestimmt werden soll. Wenn $I \leq D1$ ist, dann wird der linke Zweig L des Baums genommen und wenn $D1 < I$ ist, dann wird der rechte Zweig R genommen. Die Datenstruktur (oder die Datenwerte in der Speichereinrichtung) ist – logisch – in einer Weise organisiert, so daß der Knoten $X_2$ ein Datenwert $D_2 < D_1$ aufweisen wird, und, daß der Datenwert D3 > D1 ist. Wenn man beispielsweise einen Bereich von Daten $[0, D_{max}]$ annimmt, teilt der Datenwert, der an dem Wurzelknoten $D_{RN}$ gespeichert ist, den Baum tatsächlich in Datenwerte $(0 \ldots D_{RN}-1)$, $D_{RN}$, $(D_{RN}+1 \ldots D_{max})$ auf, wobei der zuerst erwähnte Datenwert und der zuletzt erwähnte Datenwert jeweils an Knoten zwischen den linken bzw. rechten Zweigen gefunden werden kann. Da jedoch die einzelnen Datenwerte $D_1$, $D_2$, $D_3$ in der Speichereinrichtung zufällig gespeichert sein können, ist es für die Binärbaumsuche erforderlich, daß – nach Auslesen von $D_1$ bei $A(X_1)$, $B(X_1)$ – Information bezüglich des Orts oder der Speicheradresse von $D_2$ oder $D_3$ bereitgestellt wird.

Deshalb besteht herkömmlicherweise eine Lösung darin, daß jedem Datenwert $D_1$, $D_2$, $D_3$ zwei Zeiger, d. h. zwei weitere Speicherpositionen einschließlich von Adressen, die den Ort des Datenwertes anzeigen, der größer (R) oder kleiner (L) als der ausgelesene Datenwert ist, zugeordnet (mit ihm gespeichert) wird. Während logisch die Binärbaumstruktur verwendet wird, wobei in jeder weiteren Unterebene der gespeicherte Datenwert jeweils größer oder kleiner als in der vorangehenden Unterebene (Unterbaum) in der Speichereinrichtung selbst wie in **Fig. 5** gezeigt, sein wird, besteht nur eine logische Abbildung unter Verwendung der Adressenzeiger, während im Prinzip die Datenwerte zufällig in der Speichereinrichtung gespeichert sein können.

Die Verwendung der Dateneinträge mit verschiedenen Adressenzeigern erfordert die vorherige Definition des Wurzel- oder Ursprungsknotens RN, der der erste Knoten ist, von dem ein Datenwert ausgelesen werden soll und mit dem zu suchenden Datenwert verglichen werden soll. Ein derartiger Suchalgorithmus ist in "Data Structures and Algorithm", Aho Hopcroft, Ullmann; ISBN0-201-00023-7 auf den Seiten 155 ff. beschrieben. Ganz offensichtlich besteht ein Nachteil darin, daß der Speicherplatz, der zum Speichern einer großen Vielzahl von Datenwerten benötigt wird, erfordert, daß mit jedem einzelnen Datenwert zwei weitere Zeigereinträge gespeichert werden, die die Zweige L, R anzeigen.

Während in der **Fig. 5** eine explizite Darstellung des logischen Binärbaums in der Speichereinrichtung existiert, besteht eine andere Lösung darin, daß eine Abbildung der Binärbaum-Datenstruktur auf festgelegte Adressen in der Speichereinrichtung, d. h. auf die Elemente des Matrixfelds, verwendet wird. In diesem Fall sind die Verzweigungsadressen vorher durch die Abbildung der Binärbaumknoten auf vordefinierte Stellen in der Speichereinrichtung bekannt und deshalb sind hier Zeiger nicht erforderlich und belegen keinen Speicherplatz. Da die Zeiger nicht ausgewertet werden müssen, kann die Suchzeit niedriger sein, wie in "Data Structures and algorithms"; Aho, Hopperaft, Ullmann; TSBN-0-201-00023-7; Seiten 271 ff. offenbart ist. Unter Verwendung einer derartigen impliziten Abbildung der einzelnen Knoten $X_1$, $X_2$, $X_3$ auf spezifische Stellen in der Speichereinrichtung kann die Adresse der jeweiligen "Kinder-" oder untergeordneten Knoten $X_2$, $X_3$ innerhalb eines Unterbaums (untergeordneten Baums) (!) berechnet werden. Wenn man den linken und den rechten Zweig der Spaltenadresse A darstellt, kann tatsächlich die nächste Knotenadresse in dem linken Zweig wie folgt berechnet werden:

$$A(L(X)) = 2A(X) + 0$$

wohingegen der rechte Zweig folgendermaßen berechnet wird:

$$A(R(X)) = 2A(X) + 1.$$

Eine Verwendung einer derartigen Abbildung für die vollständige Adresse < B, A > würde jedoch bedeuten, daß nach einer kurzen anfänglichen Durchquerung des Baums der höchstwertige Teil der Adresse sich bei jedem weiteren Schritt der Durchquerung ändert. Deshalb kann die Suchzeit für große Bäume, die in einem dynamischen Direktzugriffsspeicher gespeichert werden müssen, aufgrund ihrer Größen beträchtlich sein. Für echte Anwendungen werden schnellere Verfahren benötigt, die insbesondere die Anzahl von Änderungen des höchstwertigen Teils der Adresse verringern.

Das Verfahren und die Zugriffseinrichtung der Erfindung basieren auf einer Abbildung von Knoten in einem Binärbaum auf die Adressen einer Speichereinrichtung, vorzugsweise auf eines dynamisches Direktzugriffsspeichers DRAM. DRAMs offerieren normalerweise eine sehr dichte Speicherung, jedoch sind sie gegenüber zufälligen verteilten Zugriffen relativ langsam. Unter Verwendung eines spezifischen Adressierungsmodus gemäß der Erfindung können

weitaus geringere Zugriffszeiten erreicht werden. Dafür müssen von den Adressen einer Abfolge von aufeinanderfolgenden Zugriffen bestimmte Beschränkungen eingehalten werden. Die Abbildung gemäß der Erfindung stellt sicher, daß die Suche durch einen Baum immer zu einer Abfolge von Zugriffen führt, die diese Beschränkungen erfüllt. Das erreichbare Betriebsverhalten (die Zeitabnahme) steigt um ungefähr einen Faktor 10 an.

Die Adress-Bits des DRAMs sind allgemein gebräuchlich, wie voranstehend erwähnt, in zwei Teile unterteilt, die als die Zeilenadresse B und die Spaltenadresse A bezeichnet werden (siehe **Fig. 5**). Normalerweise weisen die Zeilen- und Spaltenadresse die gleiche Größe auf. Wegen der internen Implementierung des DRAM ist die Zugriffszeit signifikant kürzer, wenn aufeinanderfolgende Zugriffsvorgänge lediglich dadurch ausgeführt werden können, daß die Spaltenadresse verändert wird, wohingegen die Zeilenadresse so weit wie möglich unverändert gehalten wird. Somit ermöglicht das neuartige Verfahren und die Zugriffseinrichtung gemäß der Erfindung zum Abbilden einen Baumsuchalgorithmus, der die Kriterien erfüllt, die Zeilenadresse so weit wie möglich unverändert zu halten.

**Fig. 7** zeigt die Unterteilung eines Binärbaums in eine Anzahl von Unterbäumen gemäß der Erfindung. Der vollständige Baum umfaßt eine Anzahl von Unterbäumen, von denen fünf Unterbäume **1, 2, 3, 4, 5** gezeigt sind. In **Fig. 7** weist jeder Unterknoten einen Wurzelknoten auf, der mit "k = 1" bezeichnet ist. Genauso ist jeder Blatt- oder Endknoten eines Unterbaums mit "k = K" bezeichnet. K ist die "Tiefe" des Unterbaums, d. h. es gibt eine Anzahl von anderen Knoten, die innerhalb jedes Unterbaums nur mit gestrichelten Linien dargestellt sind. Tatsächlich umfaßt jeder dargestellte Unterbaum in **Fig. 7** somit K Ebenen, was insgesamt pro Unterbaum $2^K$ 1 Knoten entspricht.

Sobald auf einen Wurzelknoten über seine Spalten- und Zeilenadresse zugegriffen worden ist, wird eine Suche jeweils in einem Unterbaum mit einer Anzahl von Ebenen K (tatsächlich hängt die Anzahl von Ebenen K von der Größe der Zeilen in dem Speicher ab) ausgeführt. Jeder Unterbaum weist (unterste) Blatt- oder Endknoten auf, die mit einem jeweiligen Wurzelknoten des nächsten Unterbaums verbunden sind. Es sei darauf hingewiesen, daß in **Fig. 7** nur eine Teilmenge der Knoten vollständig gezeichnet ist. Beispielsweise ist der Endknoten LM des Unterbaums **1** direkt mit dem Wurzelknoten $X_1$ des Unterbaums **3** verbunden. Es existieren eine Anzahl von Zwischenknoten zwischen dem Wurzelknoten $X_1$ und seinen Endknoten $X_2$, $X_3$. Der Wurzelknoten befindet sich bei k = 1 und der Endknoten befindet sich bei k = K. Der Endknoten LM des Unterbaums **5** bildet gleichzeitig auch einen Endknoten des gesamten Binärbaums. Deshalb ist es wichtig, zwischen Wurzeln (Anfangspunkten) und Blättern (Endpunkten) des gesamten Baums und Wurzeln und Blättern der Unterbäume zu unterscheiden. In jedem Unterbaum geht die Variable k schrittweise von 1 bis K für jede Durchquerung eines Unterbaums, wiederholt, bis der Wert, der gesucht wird, gefunden wird oder ein Endknoten des gesamten Baums erreicht wird.

Die Anzahl von Ebenen in einem Unterbaum hängt von der tatsächlichen Speicher-Hardware ab. Gemäß der Erfindung wird die Abbildung der Speicherstellen in einer derartigen Weise durchgeführt, daß jede Zeile $2^K$–1 Einträge, d. h. Spalten enthält. Wie sich nachstehend noch ersehen läßt, entspricht die Suche innerhalb eines Unterbaums gemäß der Erfindung einer Suche innerhalb einer einzelnen Zeile. Die Speicherstellen entlang der Zeile (z. B. entlang des Unterbaums **1**) entsprechen (sequentiell mit zunehmender Spaltenadresse) dem Wurzelknoten, den Zwischenknoten und den Endknoten LN des spezifischen Unterbaums. Jeder ausgelesene Datenwert $D_1$, $D_2$, $D_3$ wird mit dem zu lokalisierenden Datenwert I verglichen und eine Entscheidung wird getroffen, ob der linke oder rechte Zweig innerhalb des Unterbaums genommen werden soll.

Wie in **Fig. 6** für einen herkömmlichen Binärbaum gezeigt, ist ein Baum eine spezielle Art von Graph, der keine Zyklen darin aufweist; das heißt, nach Durchqueren des Baums mit einem Schritt besteht der einzige Weg zum Zurückkehren zu dem vorangehenden Knoten darin, den gleichen Weg zurückzugehen. Dies ist exakt in Übereinstimmung mit einem natürlichen Baum. Ferner ist jeder Teil eines Baums, der selbst einen Baum bildet, ein Unterbaum. Somit enthält ein Baum allgemein viele Unterbäume.

Der gesamte Baum gemäß der Erfindung, so wie er in **Fig. 7** dargestellt ist, ist in Unterbäume in einer speziellen Weise aufgeteilt, so daß die Unterbäume ausschließlich sind und nicht mit anderen Unterbäumen verbunden sind; d. h., keiner der Unterbäume (die jeweils einen Wurzelknoten, Endknoten und Zwischenknoten, die in der Tat untereinander verbunden sein können, wie in **Fig. 9** gezeigt, enthalten) enthält irgendeinen Teil der anderen Unterbäume. Natürlich existieren viele andere Vorgehensweisen zum Bilden von Unterbäumen aus dem gesamten Baum. Diese Tatsache ist wichtig, da die Tatsache, daß der Baum in Unterbäume unterteilt ist, nicht bedeutet, daß dies so durchgeführt wird, wie in **Fig. 7** gezeigt. Somit definiert **Fig. 7** den speziellen Typ von Unterbaum-Aufteilung, die in der Erfindung durchgeführt wird.

Die Organisation der Daten innerhalb einer Zeile (eines Unterbaums) ist so, wie in **Fig. 9** gezeigt. Der Wurzelknoten RN wird immer auf die erste Spaltenadresse A = 1 abgebildet. Unter der Annahme eines Datenbereichs zwischen $D_{min}$ und $D_{max}$ teilt dann der Datenwert $D_1$, der sich an dem Wurzelknoten RN befindet, den Unterbaum in Datenwerte D = $[D_{min}, D_1-1]$, die sich an Knoten auf der linken Seite befinden, und Datenwerte D = $[D_1+1, D_{max}]$, die an Knoten auf der rechten Seite gespeichert sind, auf. Es ist wichtig darauf hinzuweisen, daß durch die Unterteilung nur die Bereiche angegeben werden, nicht die exakten Werte der Daten. Tatsächlich werden die nächsten Knoten bei k = 2 wiederum eine Unterteilung gemäß des jeweiligen Datenwerts $D_2$, $D_3$ vornehmen, die an den Knoten gespeichert sind. Wie sich der **Fig. 9** entnehmen läßt, befindet sich jedoch ein Datenwert $D_2 < D_1$ an der Spaltenadresse A' = 2A, während sich der Datenwert $D_3$ auf dem rechten Zweig an der neuen Spaltenadresse A = 2A+1 befindet. Wiederum werden die Datenwerte $D_4$, $D_5$ an Spaltenadressen 2A, 2A+1 jeweils gespeichert. Unter der Annahme, daß K = 3 ist, bilden offensichtlich die Datenwerte $D_4$, $D_5$, $D_6$, $D_7$ die Blatt- oder Endknoten des Unterbaums, wie in **Fig. 9c** angedeutet ist. Allgemein wird die Adresse eines neuen Knotens auf einem linken Zweig 2A sein und auf dem rechten Zweig wird sie 2A+1 sein. Allgemein existieren $2^K$–1 Endknoten, wohingegen offensichtlich der Datenwert DK an der Spaltenadresse $2^K$–1 gespeichert wird. Die Anzahl von Knoten beträgt $2^{K-1}$-2. Dies beschreibt die Abbildung der Knoten auf die einzelnen Spaltenadressen in einer Zeile.

Sobald jedoch ein Endknoten ($D_4$, $D_5$, $D_6$, $D_7$ für K = 3 oder $D_K$ für die $2^{K-1}$ Knoten an dem rechten Ende der Spalte) in einer Zeile erreicht wird, ist die Frage, wo sich die nächsten Wurzelknoten ($D_{4'}$, $D_{4''}$ . . . für K = 3 und $D_{K'}$, $D_{K''}$ für k = K) in dem Matrixspeicher befinden. Wie in **Fig. 9** angedeutet, befinden sie die Wurzelknoten der nächsten Unterbäume immer an der Position A = 1, jedoch sind die Wurzelknoten und die nächsten Zwischenknoten notwendigerweise und die Endknoten nicht notwendigerweise in der nächsten Zeile gespeichert. Gemäß der Erfindung befindet sich der nächste Wurzelknoten des nächsten Unterbaums an der

Spaltenposition Λ = 1 und der Zeilenposition B(L(X)) und B(R(X)). Gemäß der Erfindung wird der Datenwert $D_{4'}$, $D_{K'}$ für den linken Zweig bei

$$B(L(X)) = 2^K \cdot B(X)+2 \cdot \Lambda(X)+0\omega 2^K$$

gespeichert, wohingegen der nächste Wurzelknoten eines Unterbaums, der auf einem rechten Zweig verbunden ist, sich an der Spaltenadresse

$$B(R(X)) = 2^K \cdot B(X)+2 \cdot \Lambda(X)+1\ 2^K.$$

befindet. Wie sich der **Fig. 8** ferner entnehmen läßt, befindet sich der nächste Unterbaum (die zugehörige Zeile) nicht notwendigerweise benachbart zu dem vorangehenden Unterbaum.

Die Abbildung der Knoten des Binärbaums auf die Speicherpositionen in dem Speicher gemäß **Fig. 9** bedeutet, daß die Einträge an den Knoten eines Binärbaums gespeichert sind, der gemäß einer zunehmenden Größe von Werten in dem Baum (und somit auch innerhalb des Unterbaums die mit den Werten in **Fig. 7, 8, 9** angedeutet ist) sortiert ist.

Natürlich weist die Speicherung der einzelnen Datenwerte in einer ansteigenden Größe in dem Binärbaum (entsprechend einer spezifischen Reihenfolge in der Speicherzeile, wie in **Fig. 9** angedeutet ist) einen Einfluß auf den Suchalgorithmus auf, der in jedem Unterbaum und von jedem Unterbaum-Endknoten zu dem nächsten Unterbaum-Wurzelknoten verwendet wird.

Wie voranstehend erläutert wurde, besteht das Hauptziel darin, so wenig Zeilenänderungen wie möglich vorzunehmen, d. h. den höchstwertigen Teil der Adresse S = < B, Λ > so selten wie möglich zu ändern, da nur die Suche in der Spaltenrichtung sehr schnell ist. Das Speichern der Datenwerte für einen Unterbaum in einer Zeile gemäß einer spezifischen Reihenfolge in einer Spaltenrichtung führt zu dem Ergebnis, daß die letzten Einträge in der Spaltenrichtung die Endknoten LN jedes Unterbaums bilden müssen. Wie viele Endknoten LN (Endknoten-Speicherpositionen LN in **Fig. 8**) zugewiesen sind, hängt von der Anzahl von Ebenen innerhalb des Unterbaums ab, d. h. hängt von der Größe des Speichers in der Spaltenrichtung ab. Beispielsweise umfaßt ein Unterbaum mit K = 3 Ebenen 4 Endknoten LN an den vier am weitesten rechts liegenden Positionen in der Spaltenrichtung (siehe auch das Beispiel **Fig. 11, 12**).

Überall in der Beschreibung sei in Hinsicht auf die voranstehend erläuterte Abbildung erwähnt, daß der (logische) Ausdruck "Knoten" der (physikalischen) abgebildeten Speicherstelle in dem Speicher äquivalent ist.

Wie voranstehend erläutert sieht die vorliegende Erfindung eines Abbildung des Binärbaums in dem zweidimensionalen Speicherraum derart vor, daß jede Zeile einem Unterbaum entspricht und eine spezifische Suchstrategie innerhalb jeder Zeile oder jedes Unterbaums verwendet wird. Die Suche innerhalb eines Unterbaums wird gestoppt, wenn entweder die Übereinstimmung zwischen dem ausgelesenen und dem zu suchenden Wert festgestellt wird oder ein Endknoten LF angetroffen wird. Während die Suche innerhalb jedes Unterbaums ausgeführt wird, muß der Suchalgorithmus deshalb die Ebene verfolgen, wo er sich befindet, da eine Änderung der Zeilenadresse B nur dann ausgeführt wird, wenn an einem Endknoten angekommen wird. In der vorliegenden Erfindung wurde festgestellt, daß innerhalb jedes Unterbaums nicht nur die Suchadressen berechnet werden können, sondern es wurde auch festgestellt, daß die Adresse eines spezifischen neuen Wurzelknotens eines nächsten Unterbaums nur aus dem Vergleichsergebnis und der gegenwärtigen Adresse eines Endknotens berechnet werden kann. D.h. gemäß der Erfindung wird eine neue vollständige Adresse S = < B, A >, die die Speicheradresse eines Knotens spezifiziert, an dem ein nächster Datenwert ausgelesen werden sollte, aus der gegenwärtigen Adresse der gegenwärtig ausgelesenen Speicherstelle und auf Grundlage des Vergleichsergebnisses berechnet. Dies trifft für die Adresse eines nächsten Knotens innerhalb eines Unterbaums zu und gilt auch zum Auffinden der Speicheradresse des nächsten Wurzelknotens, des nächsten zu durchsuchenden Unterbaums.

**Fig. 10** zeigt ein Flußdiagramm gemäß einer Ausführungsform des Suchverfahrens gemäß der Erfindung. Es sei darauf hingewiesen, daß für diesen Algorithmus der vollständige Binärbaum in Unterbäume gem. **Fig. 7** aufgeteilt ist und der am weitesten links liegende Unterbaum auf der Ebene unterhalb des Wurzel-Unterbaums abgetrennt ist.

Wenn, wie voranstehend erläutert, der Speicher mit m Spalten vordefiniert ist, beträgt die Anzahl von Ebenen K = ln(m)/ln2 (oder die nächste kleinere ganze Zahl). Die Anzahl von Bits in der Spaltenadresse der tatsächlichen Hardware ist gleich K, wobei K die Anzahl von Ebenen in einem Unterbaum ist. Es sei darauf hingewiesen, daß in sämtlichen Beispielen in der **Fig. 7, 8, 9** die Spaltenadresse 0 aufgrund der Baumausbildung nicht verwendet wird. Wenn m die Gesamtanzahl von Spalten und K die Anzahl von Ebenen in einem Unterbaum spezifiziert, dann werden in der Tat nur m−1 oder – falls kleiner – $2^K$−1 Spalten der m Spalten verwendet (siehe **Fig. 7, 8, 9**).

Nach Starten des Suchverfahrens im Schritt **51** wird die Ebene des ersten Unterbaums **1** deshalb auf k = 1 gesetzt, d. h. die Suche wird an dem Wurzelknoten RN des Unterbaums **1** (siehe **Fig. 7**) gestartet. Dann muß im Schritt S3 der zu suchende Datenwert I, die Anzahl von Ebenen pro Unterbaum K sowie der Ort oder die Wurzeladresse [<B(0)Λ(0)> = <0,1> (in einer Binärnotation)] des Wurzelknotens RN eingegeben werden. Deshalb ist der Wurzelknoten RN des ersten Unterbaums **1** der Eintrittspunkt des Suchalgorithmus.

Im Schritt S4 wird der Datenwert D sowie zugeordnete Information an der gegenwärtigen Adresse < B(0), A(0) > aus dem Wurzelknoten RN des Unterbaums **1** gelesen. Im Schritt S5 wird der ausgelesene Datenwert D mit dem zu lokalisierenden Datenwert I verglichen. Wenn eine Übereinstimmung D = I im Schritt S5 festgestellt wird, dann wird die Suche im Schritt S11 beendet.

Wenn im Schritt S5 der ausgelesene Datenwert D größer als der zu suchende Datenwert I ist, bestimmt der Schritt S6, ob ein Endknoten LN erreicht ist oder nicht. Dies kann durch Vergleichen der laufenden Zahl k mit der gegebenen Anzahl von Ebenen K bestimmt werden. Wenn in dem Schritt S6 festgestellt wird, daß k < K ist, dann werden Schritt S9, S14 ausgeführt. Im Schritt S9 wird die folgenden Gleichung zum Berechnen einer neuen Adresse eines Zwischenknotens und möglicherweise eines Endknotens innerhalb des Unterbaums **1**, der entlang des linken Zweigs (I<D) liegt, berechnet:

$$B(L(X)) = B(X)$$
$$\Lambda(L(X)) = 2 \cdot \Lambda(X) + 0 \quad (1).$$

Hierbei bezeichnen B(L(X)) und A(L(X)) die neue Zeilenadresse und die neue Spaltenadresse des nächsten Unterknotens innerhalb des Unterbaums **1**. Danach wird die laufende Zahl der Tiefenebene k im Schritt S14 erhöht.

Wenn im Schritt S5 bestimmt wird, daß I>D ist, wird genauso ein rechter Zweig von dem Wurzelknoten RN genommen und wenn im Schritt S8 bestimmt wird, daß die vollständige Tiefe des Unterbaums noch nicht erreicht worden

ist, d. h. k<K, dann werden die Schritte S12, S16 analog zu den Schritten S9, S14 ausgeführt, wobei die Gleichung (2) die Speicheradresse des nächsten Eintrags oder Unterknotens innerhalb des Unterbaums **1** wie folgt definiert:

$$B(R(X)) = B(X)$$
$$A(R(X)) = 2 \cdot A(X) + 1 \quad (2).$$

Wie sich dem Schritt S5, den Schritten S6, S8 und S9, S12 entnehmen läßt, wird innerhalb des Unterbaums nur der Spaltenadressenteil A modifiziert, d. h. er wird entweder für den linken Zweig verdoppelt oder für den rechten Zweig verdoppelt und um eine weitere Spalte nach rechts bewegt. Es ist wichtig darauf hinzuweisen, daß der Zeilenadressenteil B unabhängig von der Entscheidung im Schritt S5 beibehalten wird. Wenn der ausgelesene Datenwert größer ist, dann wird ein Sprung von zweimal der Spaltenadresse ausgeführt und wenn der ausgelesene Datenwert kleiner als der zu suchende I ist, dann ist es zweimal die gegenwärtige Spaltenadresse plus eins.

Da die Schritte S9, S12 nur für k < K (K = die maximale Tiefe oder die Anzahl von Ebenen innerhalb eines Unterbaums; wobei K vordefiniert ist) ausgeführt werden, wird natürlich ein neuer Datenwert D im Schritt S4 an der neuen Adresse A, B ausgelesen, die in den Schritten S9, S12 bestimmt wird. Das Bezugszeichen R bezeichnet diesen rekursiven Durchlauf durch die einzelnen Schritte S4, S5, S6, S9, S14 bzw. S4, S5, S8, S12, S16. Wenn während des rekursiven Prozesses innerhalb eines Unterbaums der Datenwert gefunden wird (I=D) kommt der Algorithmus im Schritt S11 zum Ende.

Nach k Iterationen bestimmt der Schritt S6 oder der Schritt S8 jedoch, daß die vollständige Tiefe des zugeordneten Unterbaums erreicht ist, d. h. k=K, was bedeutet, daß eine Endknoten-Speicherposition LN erreicht worden sein muß. Unter Verwendung der Abbildung von Datenwerten, wie in **Fig.** 1c ist deutlich, daß die Endknoten LN in dem Adressenraum der Speichereinrichtung in **Fig.** 9 wenigstens $2^{K-1}$ Speicherpositionen an dem höheren Ende der Spaltenadressen A (d. h. an dem rechten Ende der Zeile) entsprechen müssen.

Wenn ein derartiger Blatt- oder Endknoten LN in einem Unterbaum erreicht wird, dann werden die folgenden Gleichungen (3), (4) im Schritt S10, S13 verwendet, um eine neue Zeilenadresse und eine neue Spaltenadresse eines Wurzelknotens eines weiteren Unterbaums zu berechnen:

$$B(L(X)) = 2^K \cdot B(X) + 2 \cdot A(X) + 0 - 2^K$$
$$A(L(X)) = 1 \quad (3)$$

$$B(R(X)) = 2^K \cdot B(X) + 2 \cdot A(X) + 1 - 2^K$$
$$A(R(X)) = 1 \quad (4).$$

Aus den Gleichungen (3), (4) kann ersehen werden, daß im Fall des Auftretens eines Endknotens die Spaltenadresse A unabhängig von der Entscheidung im Schritt S5 – auf die erste Spalte, d. h. A(L(X)) = A(R(X)) = 1 gesetzt wird. Unabhängig von der Entscheidung im Schritt S5 wird jedoch eine neue Zeilenadresse B gewählt.

Es ist interessant, auf die **Fig.** 8 und 9 hinzuweisen, in denen die Pfeile von dem Wurzelknoten LN die Berechnung gemäß der Gleichungen (3), (4) bezeichnen, daß in der Tat der nächste Wurzelknoten des nächsten Unterbaums in der Spalte A = 1 befindet, sich jedoch nicht auf dem nächsten benachbarten Unterbaum 2 befindet, wie man unter Umständen angenommen hätte. Im Gegensatz dazu führen die Gleichungen (3), (4) eine Art von Spalten-Zeilen-Abbildung durch, wobei die nächste Zeilenadresse des nächsten Wurzelknotens des nächsten zu durchsuchenden Unterbaums nicht die nächste Zeilenadresse ist, mit Ausnahme des Endknotens auf der am weitesten links liegenden Seite in dem Unterbaum (siehe z. B. die Abbildung von 13 nach 19 in den Beispiel in **Fig.** 11).

Nachdem der nächste Wurzelknoten des nächsten zu durchsuchenden Unterbaums im Schritt S13 bestimmt worden ist, wird k im Schritt S17 auf k = 1 zurückgesetzt, da offensichtlich die Suche in dem nächsten Unterbaum in der Ebene **1** beginnt, d. h. an der Wurzelknoten-Adresse. Dann wird wiederum im Schritt S4 der nächste Datenwert D ausgelesen und die Schritte S5, S6, S9, S14 oder die Schritte S5, S8, S12, S16 werden rekursiv für den nächsten Unterbaum (die neue Zeile) ausgeführt, bis entweder eine Übereinstimmung zwischen D und I festgestellt wird oder wiederum der Endknoten des neuen Unterbaums gefunden wird.

Natürlich kann es passieren, daß selbst innerhalb des nächsten Unterbaums (z. B. 4 oder 5 in **Fig.** 8) der Datenwert nicht angetroffen wird und demzufolge ein Endknoten LN erreicht wird. Wenn beispielsweise der Endknoten LN in dem Unterbaum **5** erreicht wird, dann führt weder die Gleichung (3) noch die Gleichung (4) zu einer Zeilenadresse, die noch innerhalb des Zeilenadressraums enthalten ist (wenn die Anzahl von Zeilen und Spalten die gleiche ist). Wenn die Anzahl von Zeilen größer als die Anzahl von Spalten ist, dann kann natürlich eine weitere neue Berechnung eines neuen Unterbaums ausgeführt werden. Dies bedeutet, daß weitere Unterbäume (Zeilen) verfügbar sind. In der Tat muß bereits dann, wenn der Algorithmus unter dem Unterbaum **2** oder **3** in **Fig.** 7 ankommt, die Anzahl von Bits in der Zeilenadresse zweimal so groß wie die Anzahl von Bits in der Spaltenadresse sein, da mit jeder weiteren Ebene in dem Binärbaum sich in eigentümlicher Weise die Anzahl von Knoten verdoppeln.

Wenn kein weiterer Unterbaum-Wurzelknoten bestimmt werden kann, wenn ein Endknoten in Unterbaum **4** erreicht wird, zeigt dies an, daß der Datenwert in der Binärsuche durch den Unterbaum und somit auch in dem gesamten Binärbaum nicht gefunden wurde.

Es sei darauf hingewiesen, daß vorzugsweise im Schritt S5 eine gewisse Toleranz zugelassen wird, so daß eine Übereinstimmung von D und I innerhalb eines gewissen Toleranzwerts festgestellt wird, d. h. eine Übereinstimmung wird festgestellt, wenn $|D-I| < \Delta D$ ist, wobei $\Delta D$ eine bestimmte Toleranz bezeichnet, die vordesigniert ist.

Insbesondere wird zum Erzielen eines Verhaltens gemäß der oben angegebenen Gleichung (13) den in **Fig.** 10 gezeigten Punkten A und B die folgenden in **Fig.** 17 zusätzlich gezeigten Schritte durchgeführt.

Bei einem Punkt A besteht ein erster Schritt S21A in einem Test dahingehend, ob der momentane Knoten ein Blattknoten des Gesamtbaums ist oder nicht. Ist dies nicht der Fall, so schreitet der Algorithmus zu dem in **Fig.** 10 gezeigten Schritt S6 fort. Ist jedoch der momentane Knoten ein Blattknoten des Gesamtbaums, so sind die zuletzt gesicherten Werte von D und Info (D) zu benützen, woraufhin der Algorithmus endet, wie bei den Schritten S22A und S23A in **Fig.** 17 gezeigt.

Bei einem Punkt B besteht ein erster Schritt S21B in einem Test dahingehend, ob der Momentanknoten ein Blattknoten des Gesamtbaums ist oder nicht. Ist dies nicht der Fall, so werden die Momentanwerte von D und Info (D) für einen möglichen zukünftigen Einsatz gesichert. Anschließend geht der Algorithmus zu dem in **Fig.** 10 gezeigten Schritt S8 über. Ist jedoch der Momentanknoten ein Blattknoten des Gesamtbaums, so sind die Momentanwerte von D und Info (D) zu benützen, woraufhin der Algorithmus endet, wie in den Schritten S22B und S23B in **Fig.** 17 gezeigt.

Die Fig. 11, 12 zeigen den allgemeinen Algorithmus in der Fig. 7, 8, 10 für ein spezifisches Beispiel, wenn eine Binärnotation für die Spaltenadresse A und die Zeilenadresse B verwendet wird. K = 3 bedeutet, daß 8 Spalten und (wenigstens) 8 Zeilen grundlegend in dem Speicheradressraum vorgesehen sind. Da die Gleichungen (1), (2) keine Änderung in der Spaltenadresse ergeben, wenn der allererste Wurzelknoten RN ganz oben in dem Binärbaum eine Adresse A(0),B(0) = 000,000 aufweisen würde, wird die erste Spaltenadresse A = 000 nicht verwendet. Deshalb wird der erste Knoten, in dem nachgeschaut wird, wenn der Schritt S4 in Fig. 10 zum erstenmal ausgeführt wird, immer die am weitesten links liegende Speicherstelle in dem Speicher in Fig. 11 sein, d. h. B(0)=0 und A(0)=1 für den oberen und ersten Wurzelknoten des ersten Unterbaums (d. h. für den Wurzelknoten des gesamten Binärbaums).

Unter Verwendung der binären Darstellung für die Adressennummern der Zeile und Spalte zeigt Fig. 12 den Unterbaum 1 und den Unterbaum 2, die sich in den ersten zwei Zeilen mit den Adressen (000, 001 befinden.

Die Zuordnung der Datenwerte D innerhalb der Speichereinrichtung in Fig. 3a wird gemäß Fig. 11 durchgeführt, wie allgemein unter Bezugnahme auf Fig. 7 erläutert wurde. In dem Beispiel in Fig. 11 und Fig. 12 wird angenommen, daß sämtliche Datenwerte zu einem gegebenen Intervall 0–100 gehören. Beginnend mit dem Wurzelknoten unterteilt der entsprechende Wert 56 dieses Intervall in zwei neue Intervalle, nämlich 0 55 (linker Zweig) und 57–100 (rechter Zweig). Nun werden alle Werte, die zu dem unteren Intervall 0 55 gehören, in dem linken herabhängenden Baum gespeichert, wohingegen sämtliche Werte, die zu dem höheren Intervall 57 100 gehören, in dem rechten herabhängenden Baum gespeichert werden. Dieser Prozeß wird für sämtliche niedrigeren Ebenen unter Verwendung eines neuen Intervalls auf jeder Ebene rekursiv wiederholt. Beispielsweise wird für den linken Abschnitt von dem Wurzelknoten das Intervall 0 55 verwendet. Der Wert, der an dem nächsten Knoten gefunden wird ist 27. Somit ist die nächste Unterteilung in das Intervall 0 26 und 28-55 für die zwei herabhängenden Unterbäume, die von dem Knoten herabhängen, der den Datenwert 27 enthält. Dies bedeutet, daß sämtliche Werte, die zu 0 26 gehören, in dem linken herabhängenden Unterbaum gespeichert werden und sämtliche Werte, die zu 28 55 gehören, in dem rechten herabhängenden Baum gespeichert werden.

Es sei darauf hingewiesen, daß diese Art einer Abbildung von Datenwerten wie in Fig. 12 ↔ 11 gezeigt ist, für den vollständigen Baum und für einen einzelnen Unterbaum selbst zutrifft, wobei beachtet wird, daß die Startintervalle gemäß der Anzahl von Ebenen K unterschiedlich sein können. Was in Fig. 12 bezüglich der Datenwerte gezeigt ist, ist deshalb ein Binärbaum, der sich vollständig im Gleichgewicht befindet. Dies wird genauso mit den Einträgen in jeder Zeile reflektiert, die jeweils den Knoten eines Unterbaums entsprechen.

Die wichtigste Schlußfolgerung der Abbildung ist, daß der Wert in dem Baum überhaupt nicht existieren kann, wenn ein Blatt- oder Endknoten des gesamten Baums erreicht wird, ohne daß die Werte, nach denen gesucht wird, gefunden worden sind.

Eine Multiplikation mit zwei für einen linken Zweig L bedeutet einfach eine Verschiebung des Spaltenadressenteils A nach links. Ein rechter Zweig R führt dazu, daß der Spaltenadressenteil A nach links verschoben wird und zu ihm "1" addiert wird. Somit entsprechen die einzelnen Adressen bei den Zwischenknoten bei k=2 und bei den Endknoten bei k=3 =K der bestimmten Zeile und Spalte, an der sich ein bestimmter Datenwert D befindet. Fig. 12 zeigt die Datenwerte 56, 57, 82, . . . an den jeweiligen Zwischenknoten und Endknoten, so wie dies unter Bezugnahme auf Fig. 9 diskutiert wurde.

Offensichtlich berechnet der Algorithmus in Fig. 10 eine neue Wurzelknotenadresse eines nächsten Unterbaums immer dann, wenn er einen der vier Knoten 0,100 oder 0,101 oder 0,110 oder 0,111 erreicht.

Es gibt nur einen Zweig für die Berechnung eines neuen Wurzelknotens, der nicht zugelassen ist, nämlich die Adresse 0,100 in einer Situation, bei der der ausgelesene Datenwert D=13 größer als der zu suchende Wert I ist. In diesem Fall sollte tatsächlich ein linker Zweig von der Adresse 0,100 genommen werden, d. h. die Gleichung (3) sollte verwendet werden. Jedoch würde in diesem Fall die Gleichung (3) für einen linken Zweig $B(L(X)) = 2^K(000) + 2(100) - 2^K = 0000 - 1000 - 1000 = 0000$ ergeben. Deshalb darf der erste linke Zweig, der in Fig. 7 und Fig. 12 mit "abgeschnitten" angedeutet ist, nicht genommen werden, da die Suche erneut von der Wurzelknotenadresse beginnen würde.

In der Tat weist die Suchmaschine selbst keinen Test auf, der vermeidet, daß die Durchquerung in den verbotenen (abgeschnitten) Unterbaum läuft. Jedoch kann dies sichergestellt werden, indem ein spezifischer Datenwert an dem linken Endknoten des ersten (Wurzel-) Unterbaums gespeichert wird. Das heißt, der am weitesten links liegende Knoten in dem Wurzel-Unterbaum enthält einen Teil mit einem Datenwert, der garantiert, daß der rechte Zweig und somit der nächste Unterbaum nach rechts (Datenwert 19 in Fig. 12) immer durch die Sucheinrichtung gewählt wird. Wenn die Teile in der Folge in dem Baum gespeichert werden, wie in Fig. 12 gezeigt, wird diese Bedingung automatisch erfüllt sein. Wenn der Baum abgeschnitten ist, wird auch der am weitesten links liegende Knoten in dem Wurzel-Unterbaum der am weitesten links liegende Knoten in dem gesamten Baum. Somit ist dies der Ort, an dem der Teil mit dem niedrigsten Wert gespeichert wird. Jeder andere Wert liegt auf der rechten Seite.

Wenn jedoch ein rechter Zweig genommen wird, bedeutet dies gemäß Gleichung (2) B(R(X)) = 001, daß der am weitesten links liegende Endknoten LN mit der Adresse 0,100 zu einer nächsten Endknoten-Wurzeladresse von 001, 001 führt. Wie in Fig. 12 angedeutet, entspricht dies tatsächlich dem Wurzelknoten des Unterbaums 2, der sich in der zweiten Zeile befindet. Wie sich aus Fig. 11 ersehen läßt, kann jeder der Endknoten dieses Unterbaums 1 auf neue Wurzelknoten von neuen Unterbäumen, die sich an der Position A=1 an einer spezifischen Zeilenadresse befinden, "abgebildet" werden. Wenn die Anzahl von Zeilen wenigstens die gleiche Größe wie die Größe der Spalten ist, kann sichergestellt werden, daß jeder Endknoten LN des Unterbaums 1 in eine neue Wurzelknotenadresse in der ersten Spalte "abgebildet" oder "transponiert" werden kann.

Wie sich ferner den Fig. 11, 12 entnehmen läßt, wenn die Datenwerte in den Knoten des Binärbaums gespeichert werden, stellt der mit den Gleichungen (1), (4) beschriebene Suchalgorithmus sicher, daß eine minimale Anzahl von Zeilenadressenänderungen verwendet wird, während die Speichermatrix zum Auffinden einer Übereinstimmung (innerhalb einer Toleranz) mit dem Datenwert I in einer minimalen Zeit durchsucht wird. Wenn keine Übereinstimmung an dem untersten Endknoten gefunden wird, dann wurde der Datenwert nicht gefunden.

Wenn bevorzugt eine Binärdarstellung des Spaltenadressenteils A des Zeilenadressenteils B verwendet wird, kann die Suche erleichtert werden, da eine Multiplikation mit 2 nur die Verschiebung der Adresse nach links in einem Schieberegister bedeutet.

Wie voranstehend erläutert kann die Änderung in der Zei-

lenadresse beim Durchqueren von einem Unterbaum zu dem nächsten Unterbaum durch die folgenden Schritte beschrieben werden. Zunächst wird die alte Zeilenadresse K-Schritte nach links verschoben, wobei mit Nullen aufgefüllt wird. Dann wird der Algorithmus innerhalb des Unterbaums zum Ändern der Spaltenadresse verwendet (was K+l Bits in der neuen Spaltenadresse impliziert). Dann wird das am weitesten links stehende Bit dieser Spaltenadresse fallengelassen (was K Bits in der neuen Spaltenadresse impliziert). Dann wird diese Spaltenadresse in die K 0 Bits der Zeilenadresse addiert. Schließlich wird die Spaltenadresse auf 1 gesetzt. Wie ersehen werden kann, werden bei jeder Änderung des Unterbaums K mehr Bits benötigt, um die Zeilenadresse zu halten, wie voranstehend unter Bezugnahme auf die **Fig. 7**, 8, 9, 11 und 12 erläutert wurde. Somit muß die Anzahl von Bits in der Zeilenadresse ein Vielfaches der Anzahl von Bits in der Spaltenadresse sein. Dies ist die bevorzugte Abbildung gemäß der Erfindung unter Einsatz einer Ausführungsform der Übersetzungstabellen-Speichereinheit mit Binärbaumstruktur.

Jedoch ist eine ähnliche Betrachtung für den Fall erfüllt, bei dem die Anzahl von Bits in der Zeilenadresse nicht ein Vielfaches der Anzahl von Bits in der Spaltenadresse ist. Hierbei wird zunächst der kleinste Baum aufgebaut, der die voranstehend beschriebene bevorzugte Anzahl von Zeilen und Spalten erfüllt. Dann werden sämtliche Unterbäume abgeschnitten, die Knoten **1** in dem nicht-existierenden Teil der Zeilenadresse enthalten. Somit kann auch ein Speicher verwendet werden, in dem die Anzahl von Zeilen nicht ein Vielfaches der Anzahl von Spalten ist, wenn ein Abschneiden von Zweigen verwendet wird und die Zuordnung der Daten entsprechend durchgeführt wird. Um wiederum zu vermeiden, daß der Algorithmus in verbotene Unterbäume geht, die abgeschnitten worden sind (die tatsächlich nicht vorhanden sind), kann das spezifische Speichern von Datenwerten derart verwendet werden, daß niemals eine Notwendigkeit besteht, daß ein Pfad zu dem abgeschnittenen Zweig genommen wird.

Nachstehend wird eine Hardwarerealisation einer Ausführungsform der Zugriffseinrichtung gemäß der Erfindung unter Verwendung eines derartigen Suchverfahrens, so wie es in den **Fig. 7** bis 12 erläutert wurde, unter Bezugnahme auf **Fig. 13** erläutert.

In **Fig. 13** bezeichnet DRAM die Speichereinrichtung **54** und die Zugriffseinheit umfaßt eine Vergleichseinrichtung MC **84** zum Vergleichen eines ausgelesenen Datenwerts D mit dem zu suchenden Datenwert I, um zu bestimmen, ob der ausgelesene Datenwert D größer oder kleiner als der zu suchende Datenwert I ist. Wenn eine Übereinstimmung (innerhalb einer gegebenen Toleranz ΔD) zwischen D und I in der Vergleichseinrichtung **84** gefunden wird, wird ein Übereinstimmungssignal M ausgegeben. Das Vergleichsergebnis C zeigt an, ob der Wert D größer oder kleiner als der Wert I ist. Eine Ausleseeinrichtung **82** (die nicht mit Einzelheiten dargestellt ist) liest einen Datenwert von einer gegenwärtigen Suchadresse A, B aus, die von den zwei Registern RA, RB eingegeben wird, die während der Durchquerung des Baums die Spalten- und Zeilen-Zwischenadressen halten.

Die Zugriffseinheit umfaßt ferner eine Bestimmungseinrichtung **86** einschließlich der Register RB, RA zum Bestimmen einer vollständigen Suchadresse, die als nächstes nach dem Datenwert durchsucht werden soll, auf Grundlage des Vergleichsergebnisses C und der gegenwärtigen Suchadresse A, B.

Die Bestimmungseinrichtung **86** umfaßt erste und zweite Register und eine erste und zweite Berechungsschaltung SMB, SMA zum Berechnen der nächsten Spalten- und Zeilen-Adressen A', B', die durchsucht werden sollen, in Abhängigkeit von dem Vergleichsergebnis C, der gegenwärtigen Adresse B, A und einem Steuersignal S, das von einer Zustandsmaschine SEQ ausgegeben wird.

Die Zustandsmaschine SEQ bestimmt den Suchzustand (tatsächlich überwacht sie k und führt somit eigentlich die Schritte S6, S8 in **Fig. 10** aus) während des Suchvorgangs durch die Binärbaum-Datenstruktur und bestimmt das Steuersignal S auf Grundlage des Vergleichsergebnisses. Im wesentlichen umfaßt die Zustandsmaschine SEQ einen Zähler und eine Zustands-Bestimmungseinrichtung STDM, um ein Steuersignal S auf Grundlage des internen Zustands $STATE_i$ zu erzeugen. Im wesentlichen entspricht der interne Zustand der Zustandsmaschine der gegenwärtigen Ebenennummer k, die immer dann aktualisiert wird, wenn ein neues Vergleichsergebnis C (was anzeigt, ob $D \leq I$ oder $D > I$ ist), von der Vergleichseinrichtung MC ausgegeben wird. Das heißt, der Zustand ist für den Wurzelknoten des Baums 0, 1 bis K−1 für nicht-Endknoten (außer dem Wurzelknoten) und K für Endknoten. In ähnlicher Weise ist das Steuersignal S gleich 0 für den Wurzelknoten, 1 für nicht-Endknoten (Unter-Knoten) und 2 für Endknoten. Beginnend von dem Wurzelknoten (des vollständigen Baums) mit einem Zustand 0 (für irgendeinen anderen Unterbaum-Wurzelknoten ist er 1) wird die Zustandsmaschine SEQ gemäß der Unterteilung des Unterbaums fortschreiten, wenn sie den Baum von oben nach unten durchquert. Das Signal S wird den Typ einer Adressenberechnung in SMB und SMA gemäß der Abbildung wählen.

Insbesondere führen die einzelnen Teile der Zugriffseinheit die folgenden Funktionen aus. Die erste Berechnungsschaltung SMB führt die folgende Funktion (angegeben in einer funktionellen Notation) aus, wobei K die Anzahl von Ebenen in einem Unterbaum ist:

$$B' = 0, \text{ wenn } S = 0;$$
$$B' = B, \text{ wenn } S = 1;$$
$$B' = 2^K \cdot B + 2 \cdot A + C - 2^K, \text{ wenn } S = 2; \quad (6)$$

wobei C (0 für D>I und 1 für D<I) das Vergleichsergebnis bezeichnet, B und B' die gegenwärtige und die nächste Zeilenadresse bezeichnen, S das Steuersignal bezeichnet, K die vorgegebene Anzahl von Ebenen in einem Unterbaum bezeichnet und A die gegenwärtige Spaltenadresse bezeichnet.

Die zweite Berechnungsschaltung SMA führt die folgenden Funktionen aus (angegeben in einer funktionellen Notation):

$$A' = 1, \text{ wenn } S = 0;$$
$$A' = 2 \cdot A + C, \text{ wenn } S = 1;$$
$$A' = 1, \text{ wenn } S = 2, \quad (7)$$

wobei C das Vergleichsergebnis bezeichnet und A und A' die gegenwärtige und nächste Spaltenadresse bezeichnen und S das Steuersignal bezeichnet.

Die Zustandsmaschine SEQ berechnet das Steuersignal auf Grundlage der folgenden Gleichungen (8) und (9):

$$S = 0, \text{ wenn } STATE_i = 0;$$
$$S = 2, \text{ wenn } STATE_i = K;$$
$$S = 1, \text{ wenn } 0 < STATE_i < K, \quad (8)$$

wobei eine Zustandsbestimmungseinrichtung STDM der Zustandsmaschine SEQ einen internen Zustand $STATE_i$ (d. h. den Wert k in **Fig. 10**) der Zustandsmaschine SEQ gemäß der folgenden Gleichung (9) berechnet:

$$STATE_i = 0, \text{ wenn } i = 0;$$
$$STATE_{i+1} = 2, \text{ wenn } STATE_i = 0;$$

$STATE_{t+1} = 1$, wenn $STATE_t = K$;
$STATE_{t+1} = STATE_t + 1$, wenn $0 < STATE_t < K$ (9)

wobei $STATE_t$ und $STATE_{t+1}$ den Zustand zu den Zeiten t und t+1 bezeichnet und t die Anzahl von Vergleichen bezeichnet, die von dem Beginn während der Durchquerung des vollständigen Baums gezählt werden (d. h. t ist eine Notation der Zeit oder der Zyklen seit dem Start der Suche in Einheiten von Vergleichsschritten), wobei t von 0 zu der Anzahl von Ebenen in einem vollständigen Baum geht, was im Prinzip eine beliebig hohe Zahl sein kann, aber auf Vielfache von K beschränkt ist, und wobei $STATE_t=0$ den Zustand bezeichnet, wenn der erste Vergleich an einer Speicherstelle entsprechend dem Wurzelknoten RN der Binärbaum-Datenstruktur ausgeführt wird.

Wie voranstehend erläutert verwenden das Verfahren und die Zugriffseinrichtung gemäß der Erfindung nur eine minimale Anzahl von Schritten, bevor eine Änderung in einem Unterbaum verursacht wird. Als Folge der Abbildung ändert sich nur die Spaltenadresse während einer Durchquerung innerhalb des Unterbaums. Wenn dynamische Direktzugriffsspeicher verwendet werden müssen, aufgrund der Menge der Daten, kann die Erfindung eine schnelle Suchmaschine nach einem Datenwert bereitstellen, der gemäß einer Binärbaum-Datenstruktur sortiert ist. Die Zeitreduktion beträgt eine Größenordnung. Somit kann die Erfindung in sämtlichen technischen Gebieten verwendet werden, in denen die Zeit zum Lokalisieren eines Datenwerts in einem großen Speicher kritisch ist.

Es ist zu erkennen, daß bei der obigen Beschreibung die Zeilen durch Spalten ersetzt werden können, da es von der Organisation des Speichers abhängt, welche Speicherrichtung jeweils als Zeile oder Spalte bezeichnet wird.

Sobald die oben beschriebene Zugriffseinheit **56** den Eindruck für eine Adresse in der Übersetzungstabellen-Speichereinheit **54** bestimmt hat, sind in der Umsetzunterstützungseinheit **50** die Startadresse DWA der Datenstruktur $d_j$ im Zusammenhang mit der umzusetzenden Datenvariablen verfügbar, sowie die Strukturinformation DINFO und die Startadresse Pi des zugeordneten Umsetzprogramms.

Der nächste Schritt betrifft dann die Bestimmung der Indizes in Übereinstimmung mit der umzusetzenden Datenvariablen. Derartige Indizes ermöglichen den Zugriff auf die umzusetzende Datenvariable durch das Umsetzprogramm unter Einsatz von logischen und nicht von physikalischen Adressen.

Während der Inversadreßberechnung bestimmt die Inversadressen-Berechnungseinheit **58** zunächst einen Wert

$$\alpha_{inv} = \alpha \quad DWA \quad (12).$$

Anschließend wird dieser Wert $\alpha_{inv}$ in Versatzgrößen 1 ... N aufgeteilt, die im Zusammenhang mit den Indizes einer allgemeinen Datenstruktur stehen und durch ein Umsetzprogramm mit der Startadresse Pi zum Durchführen eines Zugriffs auf die Datenvariable vor der tatsächlichen Umsetzung unter Einsatz einer logischen Adressierung benutzt werden.

**Fig. 14** zeigt die Berechnung der inversen Adresse für ein eindimensionales Feld. Hier ist der Datenstruktur $d_j$ mit der umzusetzenden Datenvariablen $\alpha$ die Adresse DWA bei dem Ausgangsindex hierfür zugeordnet. Demnach wird nach der Berechnung der Differenz zwischen $\alpha$ und DWA die Strukturinformation DINFO zum Bestimmen des Indizes der Datenvariablen benutzt, die die Länge jeder Datenvariablen in dem eindimensionalen Feld speichert. Insbesondere wird dieser Index durch Teilen der Differenz zwischen $\alpha$ und DWA mit der Länge jeder Datenvariablen in dem eindimensionalen Feld abgeleitet.

Die **Fig. 15** zeigt ein weiteres Beispiel im Zusammenhang mit der Inversadreßberechnung für ein zweidimensionales Feld oder eine Matrix. Es sei angenommen, daß die Matrix A(I, J) so dimensioniert ist, daß I einen Bereich zwischen 0 bis 3 aufweist und J einen Bereich zwischen 0 bis 6 aufweist. Der obere Teil der **Fig. 14** zeigt die Umsetz-Warteschlangeneinheit **52** zum Handhaben einer Adresse $\alpha$, die auf das Element A(2, 3) der Matrix A zeigt.

Ferner ist, wie im unteren Teil der **Fig. 14** gezeigt, die zweidimensionale logische Darstellung der Matrix A(I, J) in den Speicherpartitionen **12, 32** auf eine lineare Struktur abgebildet. Hier repräsentiert DWA eine Basisadresse der Matrix A gemäß dem Matrixelement A(0,0).

Zum Bestimmen der Indizes 2, 3 der Datenvariablen ist es erforderlich, die Basisadresse DWA zu kennen sowie die Länge jeder Datenvariablen der zweidimensionalen Matrix. Beide Größen sind in DINFO gespeichert, genauso wie der Bereich der Indizes, d. h. für den ersten Index (Zeile) zwischen 0 bis 3 und den zweiten Index (Spalte) zwischen 0 bis 6. Ist die Länge jedes Feldelements bekannt, so können die Indexwerte folgendermaßen berechnet werden:

$$k = \alpha - DWA$$
$$J = k - 7 \times INT(k/7)$$
$$I = INT(k/7) \quad (13).$$

INT(Z) ist der Integerteil von Z, d. h. INT(12/7) = 1, und die Konstante **7** wird aus dem Bereich des zweiten Indizes (Zahl der Spalten) und der Länge jedes Datenelements (1) abgeleitet. Demnach weist die Basisadresse von A einen Wert von 1000 auf, und die Adresse 1017 entspricht

$$J = 17 \quad 7 \times INT(17/7) = 3$$
$$I = INT(17/7) = 2, \quad (14)$$

d. h. die Speicheradresse 1017 entspricht der Variablen A(2, 3).

Durch Einsatz dieser Vorgehensweise läßt sich die Inversadreßberechnung auf jede Zahl von Indizes und jede Länge einzelner Feldelemente anwenden. Beträgt beispielsweise die Elementlänge zwei Speicherworte, so wären die Konstante 7 in den Formeln für I und J durch $2 \times 7 = 14$ zu ersetzen. Ist ein Zugriff auf jedes Wort der zwei Wort-Feldelemente möglich, so sind die Formeln so zu modifizieren, daß das obere und untere Wort jedes Elements dadurch berechnet wird, daß REM(Z) als Restteil von Z herangezogen wird.

Obgleich vorangehend die vorliegenden Erfindung unter Bezug auf die Übertragung von Variablen auf der Grundlagen von variablen Adressen beschrieben wurde, können auch Adressen auf einem höheren Niveau eingesetzt werden. Diese Adressierung auf höherem Niveau könnte aus der Programmeinheitsnummer und der Basisadresse der Variablen bestehen sowie dem Wert jedwedgen Indizes, sofern anwendbar. Diese Option der Erfindung würde die Umsetzhardware vereinfachen, da die Zugriffseinheit **56** einfacher wäre und die Inversadreß-Berechnungseinheit **58** nicht erforderlich wären. Jedoch wäre immer noch die Übersetzung von der Programmeinheitsnummer und der Basisadresse der Variablen zu der Programmeinheitsnummer und der Startadresse des Umsetzprogramms erforderlich. Dennoch besteht ein Vorteil dieser Umsetzvorgehensweise mit Adressierung auf höherem Niveau darin, daß die für die Umsetzung erzielbare Kapazität erhöht ist.

Bezugszeichenliste

10 Erste Prozessor- und Steuereinheit

12 Erste Speicherpartition
14 Erster Adreßbus
16 Erster Datenbus
18 Erste Datenadreßbus-Steuereinheit
20 Erste Datenadreßbus-Steuerleitung
22 Erste Aktualisierungsbus-Steuereinheit
24 Aktualisierungsbus
26 Erste Aktualisierungsbus-Steuerleitung
28 Erste Kopier/Umsetz-Datenleitung 30
30 Zweite Prozessor- und Steuereinheit
32 Zweite Speicherpartition
34 Zweiter Adreßbus
36 Zweiter Datenbus
38 Zweite Datenadreßbus-Steuereinheit
40 Zweite Datenadreßbus-Steuerleitung
42 Zweite Aktualisierungsbus-Steuereinheit
44 Zweite Aktualisierungsbus-Steuerleitung
46 Zweite Kopier/Umsetz-Datenleitung
50 Umsetzunterstützungseinheit
52 Umsetz-Warteschlangeneinheit
54 Übersetzungstabelle-Speichereinheit
56 Zugriffseinheit
58 Inversadressen-Berechnungseinheit
60 Adresseneingabeleitung
62 Erste Adressenverbindungsleitung
64 Zweite Adressenverbindungsleitung
66 Erste Rückhalteanzeigeleitung
68 Zweite Rückhalteanzeigeleitung
70 Adressenzuführleitung
72 DWA-Zuführleitung
74 DINFO-Zuführleitung
76-1 bis 76-N Versatzzuführleitungen
78 Umsetzprogramm-Adressenleitung
82 Ausleseeinheit der Zugriffseinheit
84 Vergleichseinheit der Zugriffseinheit
86 Bestimmungseinheit der Zugriffseinheit

## Patentansprüche

1. Software-Bearbeitungssystem vom partitionierten Typ, enthaltend:
   a) eine erste Partition (A) mit einer ersten Speichervorrichtung (12),
   b) eine zweite Partition (B) mit einer zweiten Speichervorrichtung (32), derart, daß
   c) die erste Partition (A) und die zweite Partition (B) mit einer Verbindungsvorrichtung (24) verbunden sind und einen Zustand einer neuen Software in einer Speichervorrichtung (12', 32) an den Zustand einer alten Software in der anderen Speichervorrichtung (32', 12) während der Ausführung der alten Software anpassen, und
   d) zum Unterstützen der Übertragung von Daten von der alten Software zu der neuen Software eine Datenumsetz-Unterstützungsvorrichtung (50) für die Ausgabe der Startadresse eines Umsetzprogramms im Zusammenhang mit umzusetzenden Datenvariablen vorgesehen ist.

2. Software-Bearbeitungssystem nach Anspruch 1, dadurch gekennzeichnet, daß die Datenumsetz-Unterstützungsvorrichtung (50) die Startadresse des Umsetzprogramms entweder an eine erste Prozessor- und Steuereinheit (10) der ersten Partition (A) oder eine zweite Prozessor- und Steuereinheit (30) der zweiten Partition (B) ausgibt.

3. Software-Bearbeitungssystem nach Anspruch 1 oder 2, dadurch gekennzeichnet, daß die Datenumsetz-Unterstützungsvorrichtung 50 ein Rückhaltesignal entweder an die erste Partition (A) oder an die zweite Partition (B) jeweils über eine erste Rückhaltesignalleitung (66) und eine zweite Rückhaltesignalleitung (68) in dem Fall ausgibt, in dem die Zahl der der Datenumsetz-Unterstützungseinrichtung (50) zugeführten Adressen für die Datenumsetzung die maximal zulässige Zahl der Adressen übersteigt.

4. Datenbearbeitungssystem nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, daß die Datenumsetz-Unterstützungsvorrichtung (50) zumindest eine logische Adresse für den Zugriff auf die umzusetzende Datenvariable auf einer logischen Ebene durch das Umsetzprogramm entweder an die erste Prozessor- und Steuervorrichtung (10) oder die zweite Prozessor- und Steuervorrichtung (30) ausgibt.

5. Software-Bearbeitungssystem nach einem der Ansprüche 2 bis 4, dadurch gekennzeichnet, daß die Datenumsetzungs-Unterstützungsvorrichtung (50) enthält:
   a) eine Übersetzungstabellen-Speichervorrichtung (54) zum Speichern von Adreßinformation und Strukturinformation für umzusetzende Datenvariable; und
   b) eine Zugriffsvorrichtung zum Auffinden eines Eintrags in der Übersetzungstabellen-Speichervorrichtung (54) für eine umzusetzende Datenvariable gemäß der Adresse ($\alpha$) der Datenvariablen und der Startadresse (DWA) der Datenstruktur im Zusammenhang mit der Datenvariablen.

6. Software-Bearbeitungssystem nach Anspruch 5, dadurch gekennzeichnet, daß die Übersetzungstabellen-Speichervorrichtung (54) für jede Adresse einer umzusetzenden Datenvariablen die Startadresse der zugehörigen Datenstruktur (DWA) speichert, sowie Strukturinformation (DINFO) und eine Startadresse eines zugeordneten Umsetzprogramms.

7. Software-Bearbeitungssystem nach Anspruch 6, dadurch gekennzeichnet, daß die Strukturinformation (DINFO) zumindest Information im Zusammenhang mit der Datenvariablenlänge und der Organisation der Datenstruktur ($d_j$) im Zusammenhang mit der umzusetzenden Datenvariablen speichert.

8. Software-Bearbeitungssystem nach einem der Ansprüche 5 bis 7, dadurch gekennzeichnet, daß die Zugriffsvorrichtung (56) den Eintrag in der Übersetzungstabellen-Speichervorrichtung (54) gemäß

$$v_i = \{v_i \varepsilon V_s \mid DWA_i \leq \alpha \wedge \alpha < DWA_{i+1}\}$$

bestimmt.

9. Software-Bearbeitungssystem nach einem der Ansprüche 5 bis 8, dadurch gekennzeichnet, daß die Zugriffsvorrichtung (56) einen Eintrag in der Übersetzungstabellen-Speichervorrichtung (54) durch Ausführen einer binären Suche bestimmt.

10. Software-Bearbeitungssystem nach Anspruch 9, dadurch gekennzeichnet, daß die Zugriffsvorrichtung (56) gemäß einem der Ansprüche 21 bis 31 realisiert ist.

11. Software-Bearbeitungssystem nach einem der Ansprüche 5 bis 10, dadurch gekennzeichnet, daß die Datenumsetzungs-Unterstützungsvorrichtung (50) ferner eine Inversadressen-Berechnungsvorrichtung (58) enthält, die die aus der Übersetzungstabellen-Speichervorrichtung (54) ausgelesene Information zum Identifizieren einer umzusetzenden Variablen auf einem logischen Adressierungsniveau benützt.

12. Software-Bearbeitungssystem nach Anspruch 11,

dadurch gekennzeichnet, daß die Inversadressen-Berechnungsvorrichtung (58) zunächst die Differenz zwischen der der Datenumsetz-Unterstützungsvorrichtung (50) zugeführten Adressen (α) und der Startdresse ($DWA_i$) der identifizierten Datenstruktur ($d_j$) berechnet und zudem die über die Übersetzungstabellen-Speichervorrichtung (54) bereitgestellte Strukturinformation (DINFO) zum Ableiten mindestens eines logischen Adreßversatzes gemäß der umzusetzenden Adresse benützt.

13. Datenumsetz-Unterstützungsvorrichtung, enthaltend:

a) eine Übersetzungstabellen-Speichervorrichtung (54) zum Speichern von Adreßinformation und Strukturinformation für umzusetzende Datenvariable; und

b) eine Zugriffsvorrichtung (56) zum Auffinden eines Eintrags in der Übersetzungstabellen-Speichervorrichtung (54) für ein Datenvariablen-Umsetzprogramm gemäß der Adresse (α) der Datenvariablen

14. Datenumsetz-Unterstützungsvorrichtung nach Anspruch 13, dadurch gekennzeichnet, daß die Übersetzungstabellen-Speichervorrichtung (54) für jede Adresse einer umzusetzenden Datenvariablen eine Startadresse der zugehörigen Datenstruktur (DWA) speichert, sowie Strukturinformation (DINFO) und eine Startadresse eines zugeordneten Umsetzprogramms.

15. Datenumsetz-Unterstützungsvorrichtung nach Anspruch 14, dadurch gekennzeichnet, daß die Strukturinformation (DINFO) zumindest Information im Zusammenhang mit der Datenvariablenlänge und der Organisation der Datenstruktur ($d_j$) im Zusammenhang mit der umzusetzenden Variablen enthält.

16. Datenumsetz-Unterstützungsvorrichtung nach einem der Ansprüche 13 bis 15, dadurch gekennzeichnet, daß die Zugriffsvorrichtung (56) den Eintrag in der Datentabellen-Speichervorrichtung (54) gemäß

$$v_i = \{v_i \in V_s \mid DWA_i \leq \alpha \wedge \alpha < DWA_{i+1}\}$$

bestimmt.

17. Datenumsatz-Unterstützungsgerät nach einem der Ansprüche 13 bis 16, dadurch gekennzeichnet, daß eine Zugriffsvorrichtung (56) zum Auffinden eines Eintrags in der Übersetzungstabellen-Speichervorrichtung (54) durch Ausführen einer binären Suche vorgesehen ist.

18. Datenumsetz-Unterstützungsvorrichtung nach einem der Ansprüche 13 bis 17, dadurch gekennzeichnet, daß die Zugriffsvorrichtung (56) gemäß einem der Ansprüche 21 bis 31 realisiert ist.

19. Datenumsetz-Unterstützungsvorrichtung nach einem der Ansprüche 13 bis 18, dadurch gekennzeichnet, daß sie zum Anwenden der aus der Übersetzungstabellen-Speichervorrichtung (54) ausgelesenen Information für die Identifizierung einer umzusetzenden Datenvariablen auf einem logischen Adressierungsniveau angepaßt ist.

20. Datenumsetz-Unterstützungsvorrichtung nach einem der Ansprüche 13 bis 19, dadurch gekennzeichnet, daß die Inversadressen-Berechnungsvorrichtung (58) zunächst die Differenz zwischen der der Datenumsetz-Unterstützungsvorrichtung (50) zugeführten Adresse (α) und der Startadresse (DWA) der identifizierten Datenstruktur ($d_j$) bestimmt und ferner die über die Übersetzungstabellen-Speichervorrichtung (54) bereitgestellte Strukturinformation (DINFO) zum Ableiten mindestens eines Adressenversatzes auf logischem Niveau gemäß der umzusetzenden Datenvariablen benützt.

21. Zugriffseinrichtung für eine Speichereinrichtung zum Bestimmen der Speicheradresse (B(X), A(X)) eines vorgegebenen Datenwerts (D) in einer Speichereinrichtung, wobei die Datenwerte (D) an vorgegebenen Speicheradressen (B(X), A(X)) gemäß einer Binärbaum-Datenstruktur von Knoten, Zweigen, Unterbäumen und Blättern gespeichert sind, umfassend:

a) eine Ausleseeinrichtung (R) zum Auslesen eines Datenwerts an einer vorgegebenen gegenwärtigen Suchadresse (B(X), A(X)) der Speichereinrichtung;

b) eine Vergleichseinrichtung (MC) zum Vergleichen des ausgelesenen Datenwerts (D) mit dem zu suchenden Datenwert (T), um zu bestimmen, ob der ausgelesene Datenwert (D) größer oder kleiner als der zu suchende Datenwert (I) ist; und

c) eine Bestimmungseinrichtung (SEQ, SMB, SNA, RB, RA) zum Bestimmen einer vollständigen nächsten Suchadresse (B, A), die nach dem Datenwert durchsucht werden soll, auf Grundlage des Vergleichsergebnisses (C) und der gegenwärtigen Suchadresse (B(X), A(X)); wobei

d) die Ausleseeinrichtung, die Vergleichseinrichtung und die Bestimmungseinrichtung das Auslesen, das Vergleichen und das Bestimmen rekursiv ausführen, bis der ausgelesene Datenwert (D) mit dem zu suchenden Datenwert (I) innerhalb einer vorgegebenen Toleranz übereinstimmt (M).

22. Zugriffseinrichtung nach Anspruch 21, dadurch gekennzeichnet, daß die Datenwerte in der Speichereinrichtung in einer Matrixanordnung von Zeilen und Spalten gespeichert sind, wobei jedem Datenwert D eine Spaltenadresse (A) und eine Zeilenadresse (B) zugeordnet ist.

23. Zugriffseinrichtung nach Anspruch 22, dadurch gekennzeichnet, daß die Bestimmungseinrichtung umfaßt:

- ein erstes und ein zweites Register (RB; RA) zum Halten einer nächsten Zeilen- und Spaltenadresse (B', A');
- eine erste und eine zweite Berechnungsschaltung (SMB, SMA) zum Berechnen der nächsten Spalten- und Zeilenadresse (B', A'), die durchsucht werden sollen, in Abhängigkeit von dem Vergleichsergebnis (C), der gegenwärtigen Adresse (B, A) und einem Steuersignal (S); und
- eine Zustandsmaschine (SEQ) zum Bestimmen eines Suchzustands ($STATE_i$) während des Durchsuchens der Binärbaumdatenstruktur und zum Bestimmen des Steuersignals auf Grundlage des Vergleichsergebnisses (C)

24. Zugriffseinheit nach Anspruch 22, dadurch gekennzeichnet, daß die Datenwerte in der Speichereinrichtung in einer Binärbaum-Datenstruktur mit einer vorgegebenen Anzahl von Ebenen K in jedem Unterbaum gespeichert sind, wobei die Zustandsmaschine (SEQ) einen Zähler (CNT) umfaßt, der die Anzahl von Vergleichen (t) zählt, die von der Vergleichseinrichtung (MC) ausgeführt werden.

25. Zugriffseinrichtung nach Anspruch 23, dadurch gekennzeichnet, daß das Vergleichsergebnis (C) von der Vergleichseinrichtung (MC) gemäß der folgenden Gleichung (5) berechnet wird:

$C = 0$, falls $I < D$;
$C = 1$, falls $I \geq D$; (5)

wobei C das Vergleichsergebnis bezeichnet, I den zu suchenden Datenwert bezeichnet und D den ausgelesenen Datenwert bezeichnet.

26. Zugriffseinrichtung nach Anspruch 25, dadurch gekennzeichnet, daß die erste Berechnungsschaltung (SMB) die nächste Spaltenadresse (B') gemäß der folgenden Gleichung (6) berechnet:

$B' = 0$, wenn $S = 0$;
$B' = B$, wenn $S = 1$;
$B' = 2^K \cdot B + 2 \cdot A + C - 2^K$, wenn $S = 2$; (6)

wobei C das Vergleichsergebnis bezeichnet, B und B' die gegenwärtige und die nächste Zeilenadresse bezeichnet, S das Steuersignal bezeichnet, K die vorgegebene Anzahl von Ebenen in einem Unterbaum (Bits in der Spaltenadresse) bezeichnet, und A die gegenwärtige Spaltenadresse bezeichnet.

27. Zugriffseinrichtung nach Anspruch 25, dadurch gekennzeichnet, daß die zweite Berechnungsschaltung (SMA) die nächste Spaltenadresse (A') gemäß der folgenden Gleichung (7) berechnet:

$A' = 1$, wenn $S = 0$;
$A' = 2 \cdot A + C$, wenn $S = 1$;
$A' = 1$, wenn $S = 2$, (7)

wobei C das Vergleichsergebnis bezeichnet, A und A' die gegenwärtige und die nächste Spaltenadresse bezeichnen und S das Steuersignal bezeichnet.

28. Zugriffseinrichtung nach Anspruch 23, dadurch gekennzeichnet, daß die Zustandsmaschine (SEQ) das Steuersignal (S) auf der Grundlage der folgenden Gleichung (8) berechnet:

$S = 0$, wenn $STATE_i = 0$;
$S = 2$, wenn $STATE_i = K$;
$S = 1$, wenn $0 < STATE_i < K$; (8)

wobei eine Zustandsbestimmungseinrichtung (STDM) der Zustandsmaschine (SEQ) einen internen Zustand $STATE_i$ der Zustandsmaschine (SEQ) gemäß der folgenden Gleichung (9) berechnet:

$STATE_i = 0$, wenn $t = 0$;
$STATE_{i+1} = 2$, wenn $STATE_i = 0$;
$STATE_{i+1} = 1$, wenn $STATE_i = K$;
$STATE_{i+1} = STATE_i + 1$, wenn $0 < STATE_i < K$ (9)

wobei $STATE_i$ und $STATE_{i+1}$ den Zustand zu Zeiten t und t + 1 bezeichnet und t die Anzahl von Vergleichen bezeichnet, die von dem Zähler (CNT) gezählt werden, wobei $0 < t < K \cdot N$ gilt und N die Anzahl von Unterbaum-Ebenen des vollständigen Binärbaums darstellt und wobei $STATE_i=0$ den Zustand bezeichnet, wenn der erste Vergleich an einer Speicherstelle entsprechend dem Wurzelknoten der Binärbaum-Datenstruktur ausgeführt wird.

29. Zugriffseinrichtung nach Anspruch 21, dadurch gekennzeichnet, daß die Ausleseeinrichtung (R) Information ausliest, die in der Speichereinrichtung im Zusammenhang mit dem übereinstimmenden Datenwert gespeichert ist.

30. Zugriffseinrichtung nach Anspruch 21, dadurch gekennzeichnet, daß die Speichereinrichtung ein DRAM oder ein Cache-Speicher ist.

31. Zugriffseinrichtung nach Anspruch 30, dadurch gekennzeichnet, daß die Speichereinrichtung einen Zeilenadressenteil und einen Spaltenadressenteil aufweist und der Spaltenadressenteil zum Wählen eines Eintrags der sequentiellen Dateneinträge innerhalb einer spezifischen Cache-Zeile verwendet wird und der Zeilenadressenteil zum Wählen einer spezifischen Cache-Zeile verwendet wird.

32. Verfahren zum Bestimmen der Speicheradresse (B(X), A(X)) eines vorgegebenen Datenwerts (I) in einer Speichereinrichtung, in der Datenwerte (D) an vorgegebenen Speicheradressen (B(X), A(X)) gemäß einer Binärbaum-Datenstruktur von Knoten (X), Zweigen, Unterbäumen und Blättern gespeichert sind, umfassend die folgenden Schritte:

a) Auslesen eines Datenwerts (D) an einer gegenwärtigen Suchadresse (B(X), A(X)) aus der Speichereinrichtung;

b) Vergleichen des ausgelesenen Datenwerts (D) mit dem zu suchenden Datenwert (I) um zu bestimmen, ob der ausgelesene Datenwert (D) größer oder kleiner als der zu suchende Datenwert (I) ist; und

c) Bestimmen einer vollständigen nächsten Suchadresse (BV, A'), die nach dem Datenwert durchsucht werden soll, auf Grundlage des Vergleichsergebnisses (C) und der gegenwärtigen Adresse (B(X), A(X)); wobei

d) die Schritte a)-c) rekursiv ausgeführt werden, bis der ausgelesene Datenwert (D) mit dem zu suchenden Datenwert (I) innerhalb einer vorgegebenen Toleranz (AD) übereinstimmt.

33. Verfahren nach Anspruch 32, dadurch gekennzeichnet, daß die Datenwerte in einer Binärbaum-Datenstruktur mit einer vorgegebenen Anzahl K von Ebenen in jedem Unterbaum gespeichert werden, wobei in dem Schritt c) die nächste Adresse auf Grundlage des Vergleichsergebnisses (C), der gegenwärtigen Suchadresse und auch der vorgegebenen Anzahl von Ebenen (K) in jedem Unterbaum bestimmt wird.

34. Verfahren nach Anspruch 32, dadurch gekennzeichnet, daß die Datenwerte (D) in einem Bereich eines niedrigsten Datenwerts und eines höchsten Datenwerts sind, wobei ein Mittendatenwert, der dem Mittenwert des Bereichs entspricht, an einer vorgegebenen Wurzeladresse (B(0), A(0) = <0,1>) gespeichert ist, wobei, wenn der Schritt a) zum erstenmal ausgeführt wird, die Wurzeladresse als die gegenwärtige Adresse verwendet wird.

35. Verfahren nach Anspruch 32, dadurch gekennzeichnet, daß die Datenwerte in der Speichereinrichtung in einer Matrixanordnung von Zeilen und Spalten gespeichert werden, wobei jedem Datenwert (D) eine Spaltenadresse (A) und eine Zeilenadresse (B) zugeordnet wird.

36. Verfahren nach Anspruch 32 und 35, dadurch gekennzeichnet, daß wenn das Vergleichsergebnis (C) im Schritt b) anzeigt, daß der ausgelesene Datenwert größer als der zu suchende Datenwert (I) ist ("linker Zweig"), die Adresse im Schritt c) durch die folgende Gleichung (1) berechnet wird:

$B(L(X)) = B(X)$
$A(L(X)) = 2 \cdot A(X) + 0$ (1)

wobei X den gegenwärtigen Knoten bezeichnet, der von der gegenwärtigen Adresse (B(X), A(X)) definiert

ist, A(X) und B(X) die Spaltenadresse und die Zeilenadresse des gegenwärtigen Knotens X bezeichnen, L(X) den nächsten Knoten bezeichnet, an dem ein Datenwert gespeichert wird, der kleiner als der an dem gegenwärtigen Endknoten X gespeicherte Datenwert ist, und A(L(X)) und B(L(X)) die Spaltenadresse und die Zeilenadresse des nächsten Knotens L(X) bezeichnen.

37. Verfahren nach Anspruch 32, und 35, dadurch gekennzeichnet, daß wenn das Vergleichsergebnis im Schritt b) anzeigt, daß der ausgelesene Datenwert kleiner als der Datenwert (I) ist, der gesucht werden soll ("rechter Zweig"), die nächste Adresse im Schritt c) mit der folgenden Gleichung (2) berechnet wird:

$$B(R(X)) = B(X)$$
$$A(R(X)) = 2 \cdot A(X) + 1 \quad (2)$$

wobei X den gegenwärtigen Knoten definiert, der von der gegenwärtigen Adresse (B(X), A(X)) definiert ist, A(X) und B(X) die Spaltenadresse und die Zeilenadresse des gegenwärtigen Knotens X bezeichnen, R(X) den nächsten Knoten bezeichnet, an dem ein Datenwert gespeichert ist, der größer als der an dem gegenwärtigen Knoten X gespeicherte Datenwert ist und A(R(X)) und B(R(X)) die Spaltenadresse und die Zeilenadresse des nächsten Knotens (R(X) bezeichnen.

38. Verfahren nach Anspruch 32, 33 oder 35, dadurch gekennzeichnet, daß wenn das Vergleichsergebnis im Schritt b) anzeigt, daß der ausgelesene Datenwert größer als der Datenwert (I) ist, der gesucht werden soll ("Endpunkt links"), die nächste Adresse im Schritt c) von der folgenden Gleichung (3) berechnet wird:

$$B(L(X)) = 2^K \cdot B(X) + 2 \cdot A(X) + 0 \quad 2^K$$
$$A(L(X)) = 1 \quad (3)$$

wobei X den gegenwärtigen Knoten definiert, der von der gegenwärtigen Adresse (B(X), A(X)) definiert ist, A(X) und B(X) die Spaltenadresse und die Zeilenadresse des gegenwärtigen Knotens X bezeichnen, L(X) den nächsten Knoten bezeichnet, an dem ein Datenwert gespeichert ist, der kleiner als der an dem gegenwärtigen Knoten X gespeicherte Datenwert ist und A(L(X)) und B(L(X)) die Spaltenadresse und die Zeilenadresse des nächsten Knotens L(X) bezeichnen und $2^K$ die Anzahl von Spalten in der Speichereinrichtung ist.

39. Verfahren nach Anspruch 32, 33 oder 35, dadurch gekennzeichnet, daß wenn das Vergleichsergebnis im Schritt b) anzeigt, daß der ausgelesene Datenwert kleiner als der Datenwert (I) ist, der gesucht werden soll ("rechter Endpunkt"), die nächste Adresse im Schritt c) mit der folgenden Gleichung (4) berechnet wird:

$$B(R(X)) = 2^K \cdot B(X) + 2 \cdot A(X) + 1 - 2^K$$
$$A(R(X)) = 1 \quad (4)$$

wobei X den gegenwärtigen Knoten bezeichnet, der von der gegenwärtigen Adresse (B(X), A(X)), A(X) und B(X) die Spaltenadresse und die Zeilenadresse des gegenwärtigen Knotens X bezeichnen, R(X) den nächsten Knoten bezeichnet, an dem ein Datenwert gespeichert ist, der größer als der an dem gegenwärtigen Knoten X gespeicherte Datenwert ist, und A(R(X)) und B(R(X)) die Spaltenadresse und die Zeilenadresse des nächsten Knotens R(X) bezeichnen und $2^K$ die Anzahl von Spalten in der Speichereinrichtung ist.

40. Verfahren nach einem der Ansprüche 33 und 38 oder 33 und 39, dadurch gekennzeichnet, daß die nächste Adresse gemäß der Gleichung (3) oder (4) jeweils berechnet wird, wenn die Anzahl von Vergleichen, die im Schritt b) durchgeführt werden, einem Vielfachen der Anzahl von Ebenen in einem Unterbaum (K) gleicht.

41. Verfahren nach einem der Ansprüche 33 und 36 oder 33 und 37, dadurch gekennzeichnet, daß die nächste Adresse gemäß der Gleichung (1) oder (2) jeweils berechnet wird, wenn die Anzahl von Vergleichen, die im Schritt b) ausgeführt werden, nicht der Anzahl von Ebenen in einem Unterbaum (K) gleicht.

42. Verfahren nach Anspruch 32, dadurch gekennzeichnet, daß nach dem Schritt d) Information, die im Zusammenhang mit dem übereinstimmenden Datenwert gespeichert ist, aus der Speicherstelle mit der gegenwärtigen Adresse ausgelesen wird.

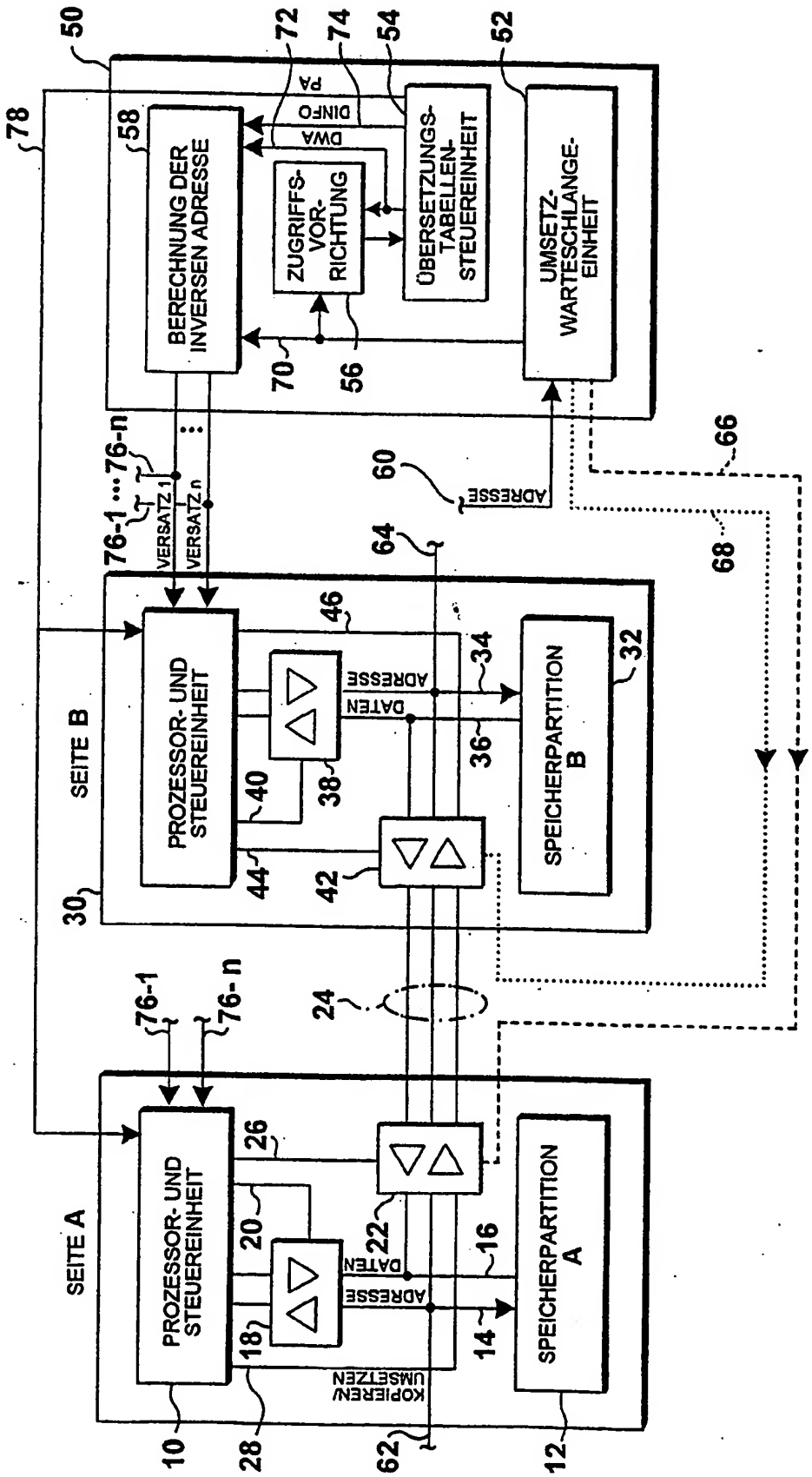
43. Verfahren nach Anspruch 32, dadurch gekennzeichnet, daß die Speichereinrichtung ein DRAM oder ein Cache-Speicher ist.

Hierzu 17 Seite(n) Zeichnungen

- Leerseite -

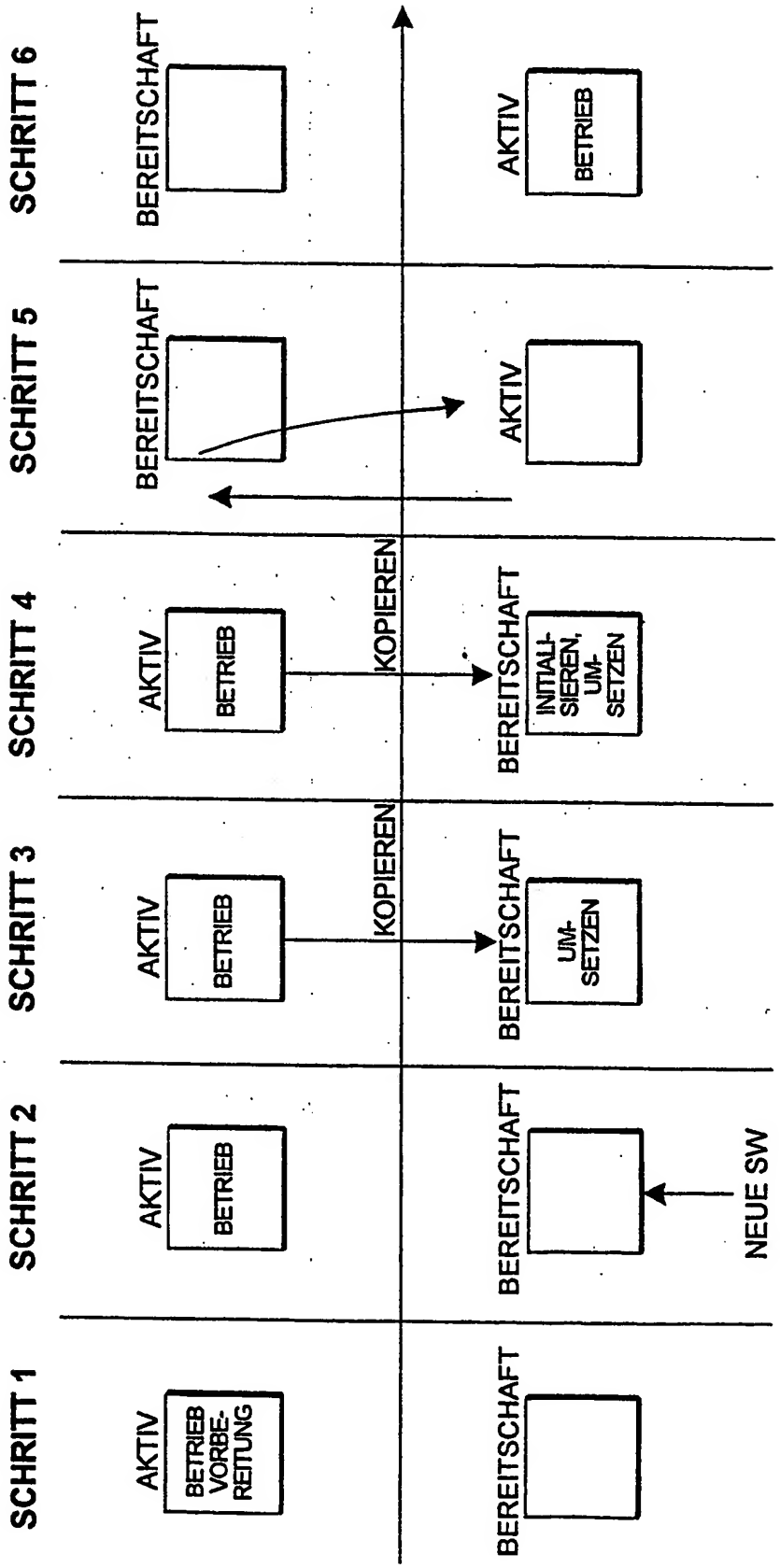
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.1



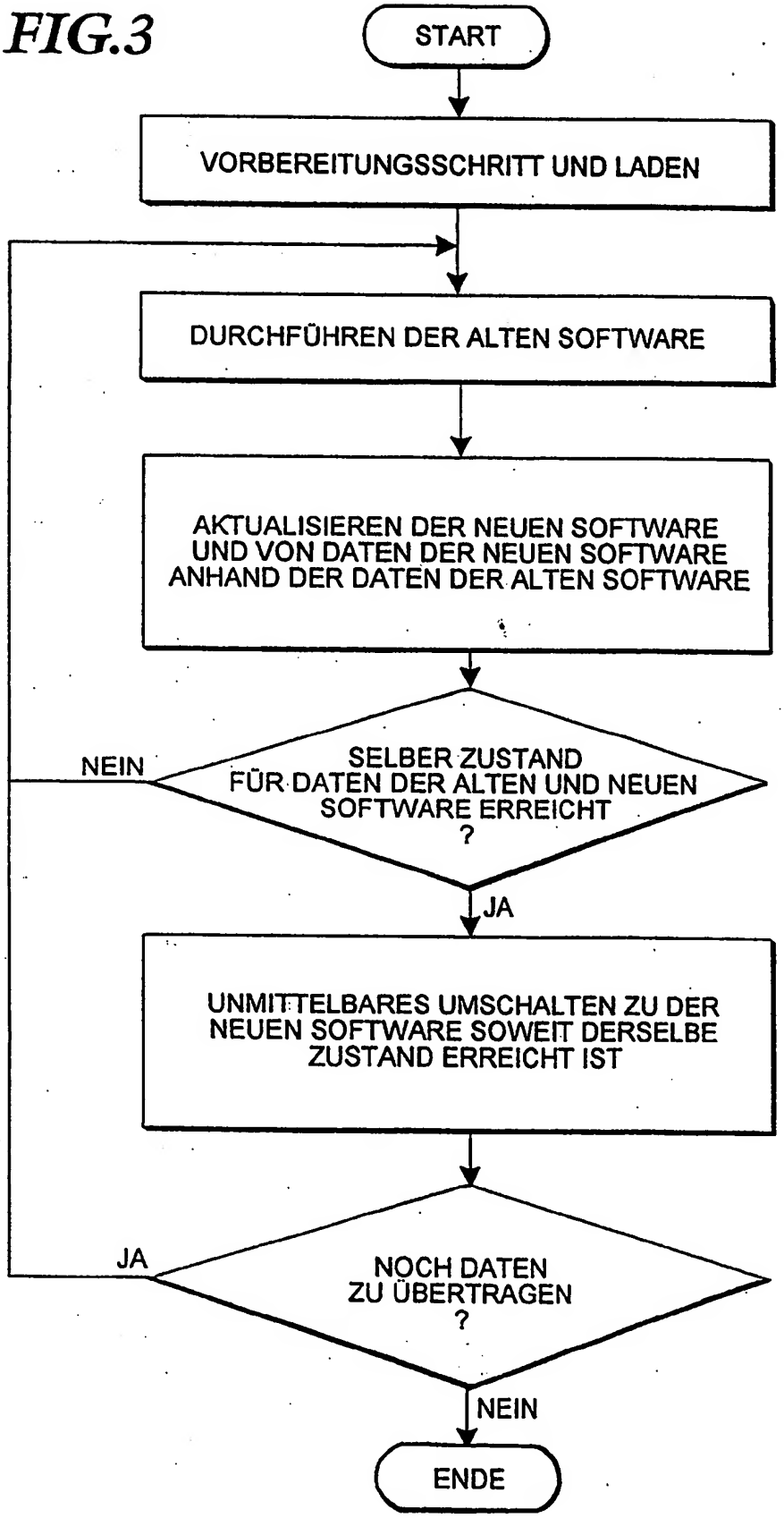
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.2



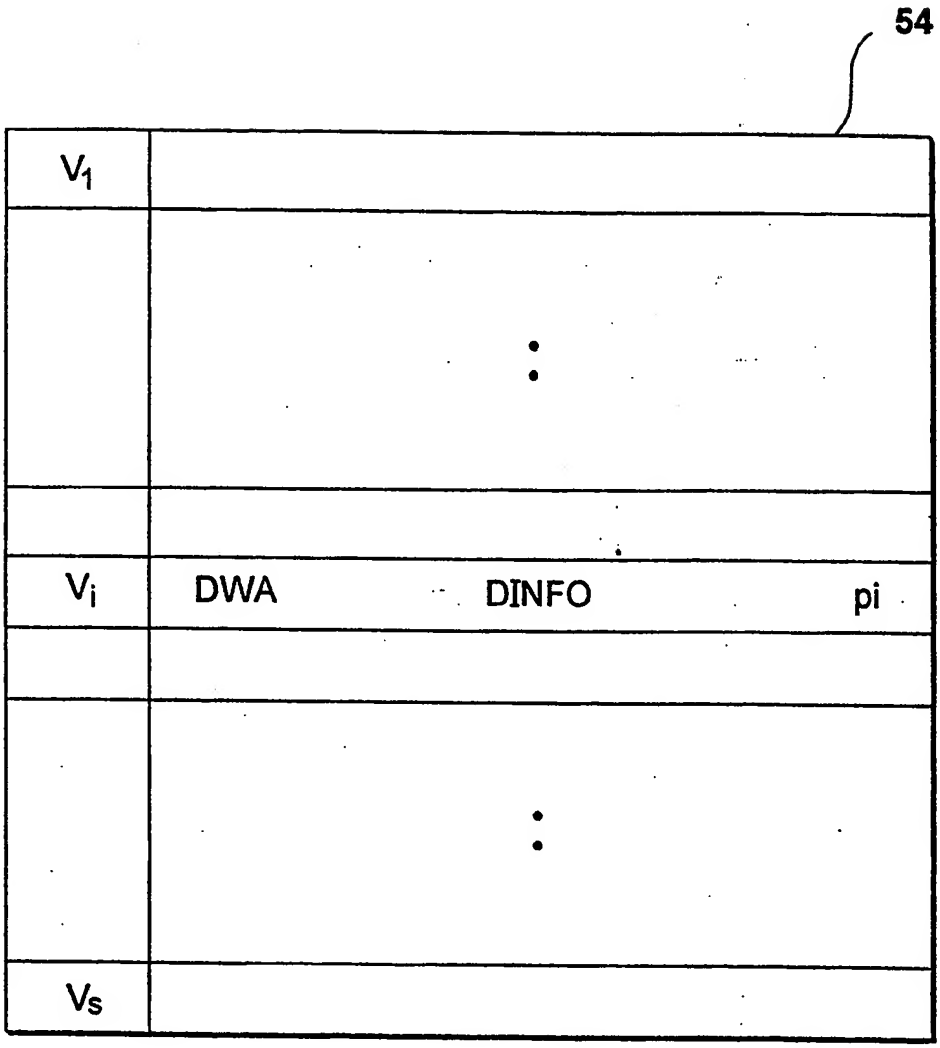
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

*FIG.3*



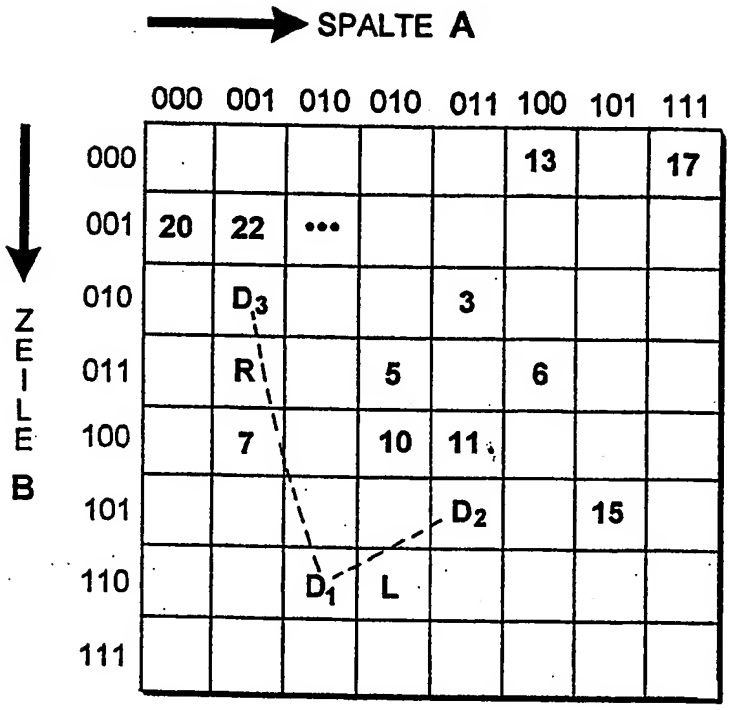
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

*FIG.4*



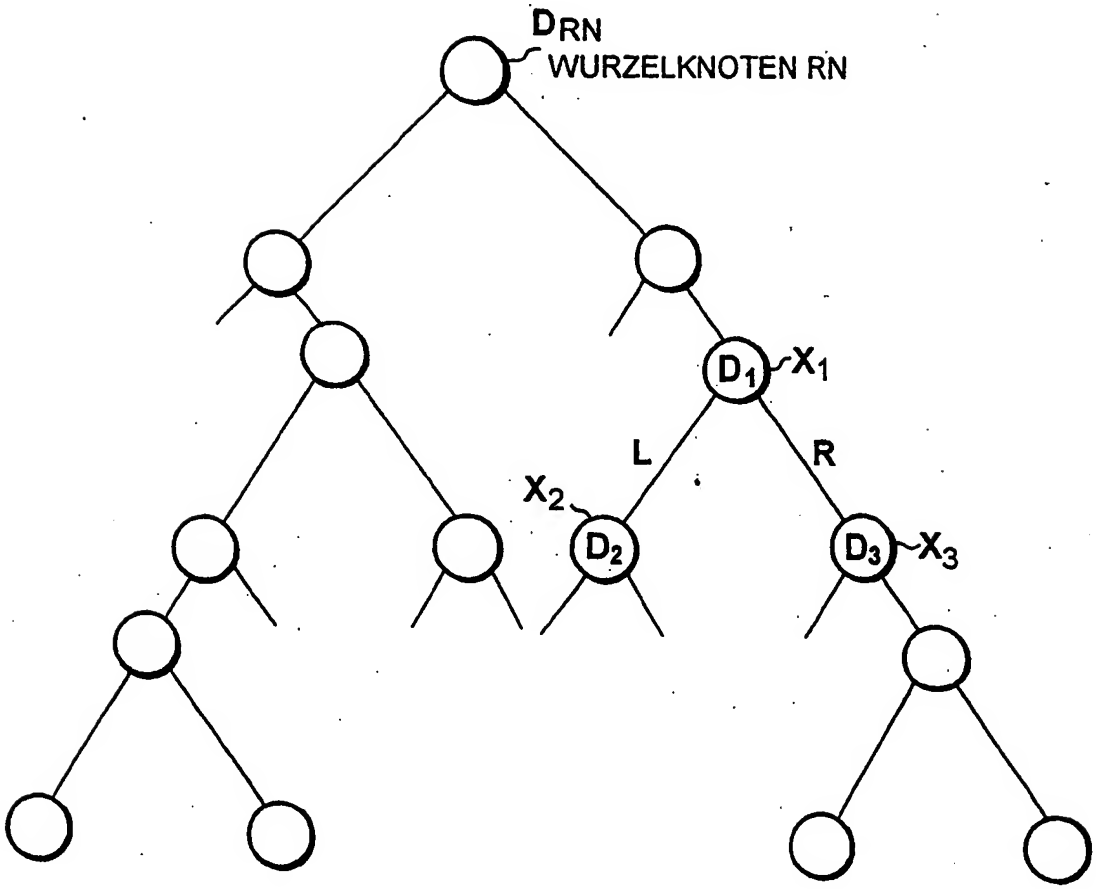
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

# FIG.5



Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

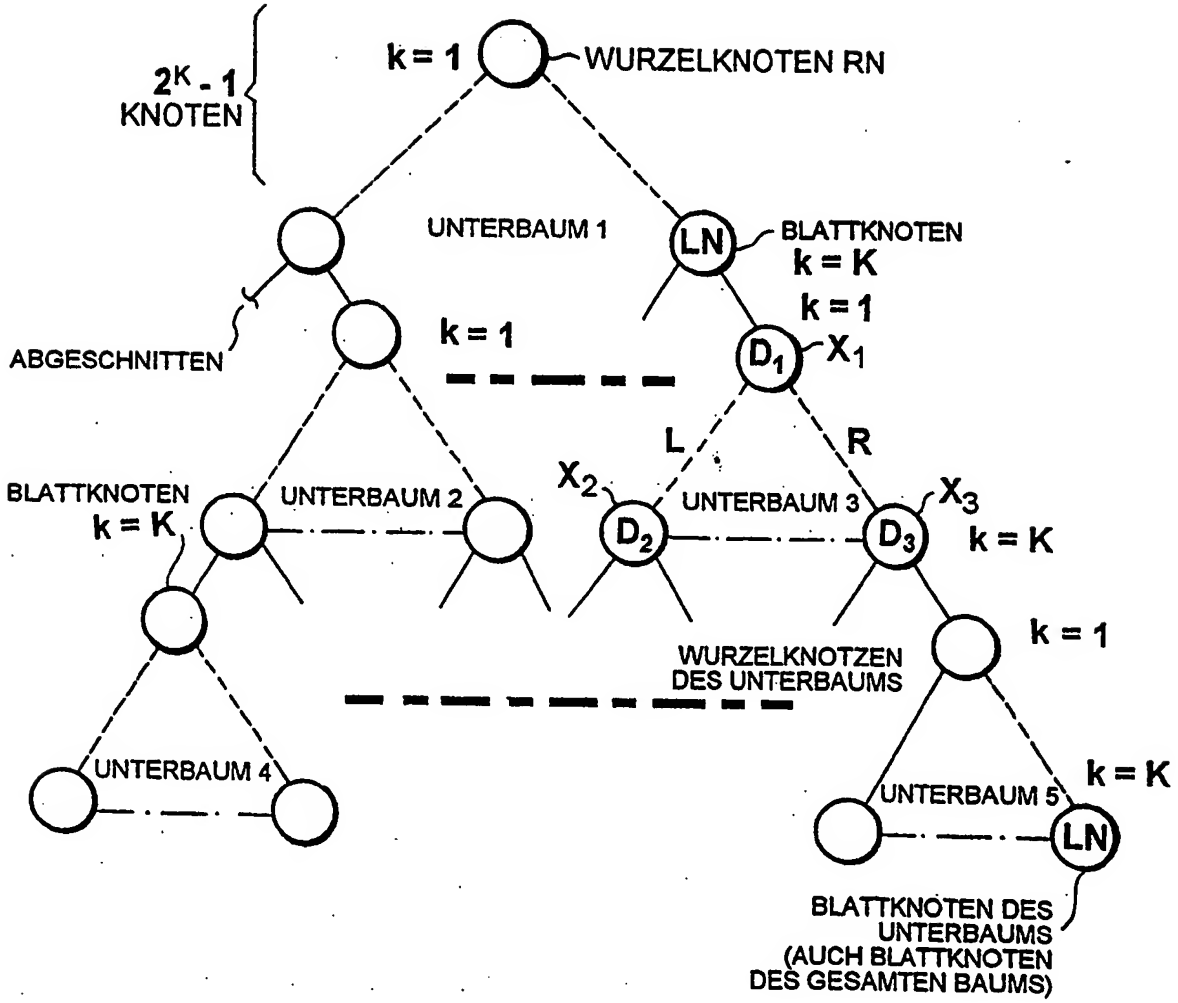
# FIG.6




**BINÄRBAUM - DATENSTRUKTUR**

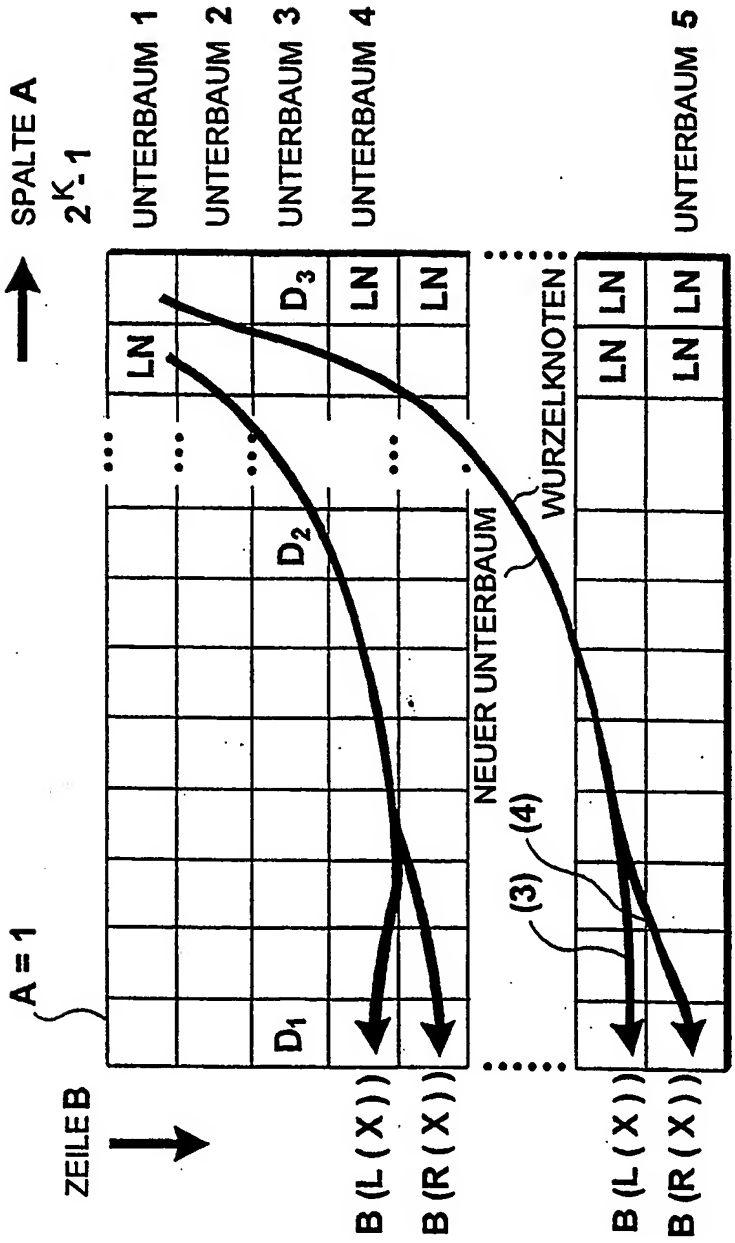
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

*FIG.7*



Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.8



Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

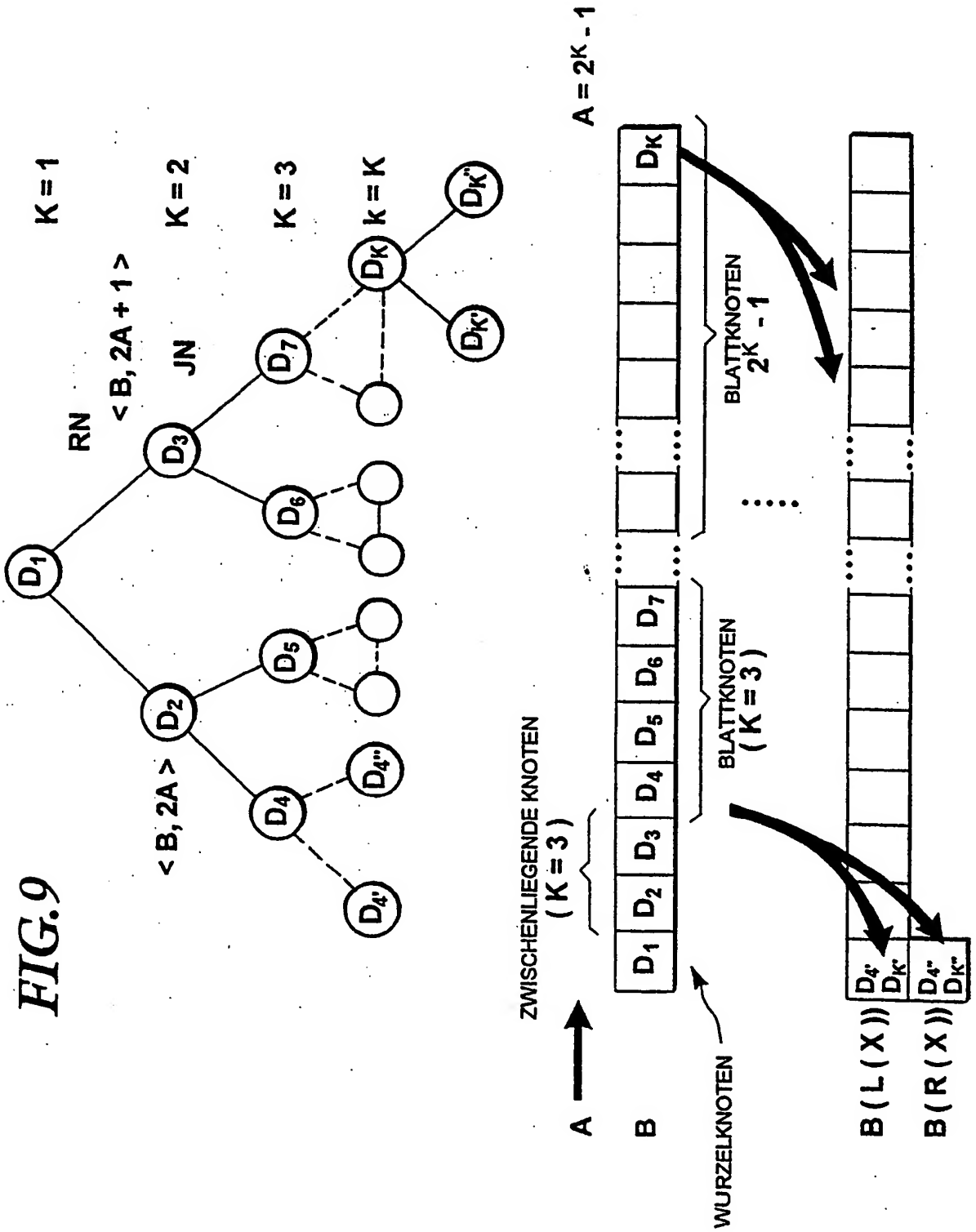



FIG.9

Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

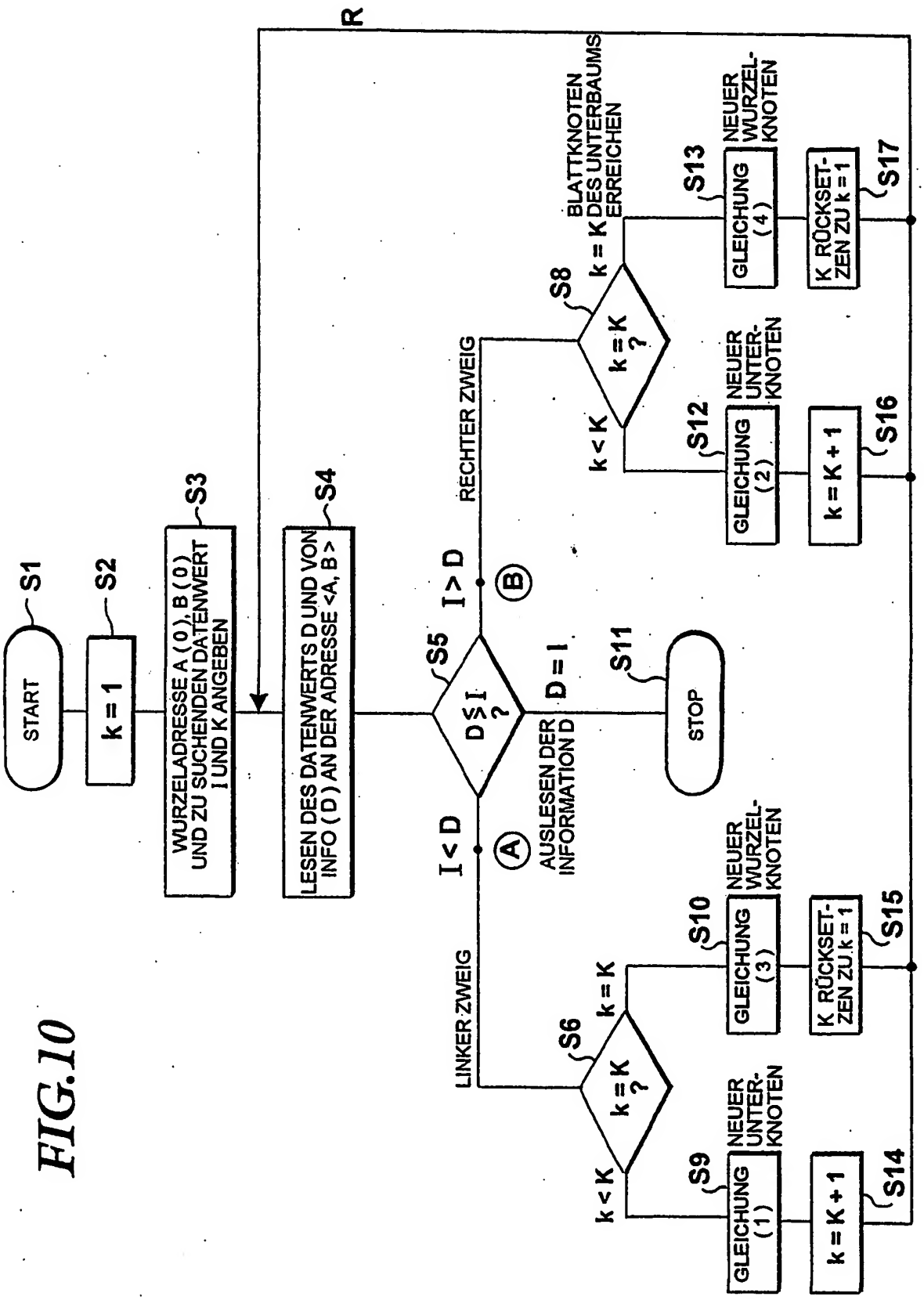
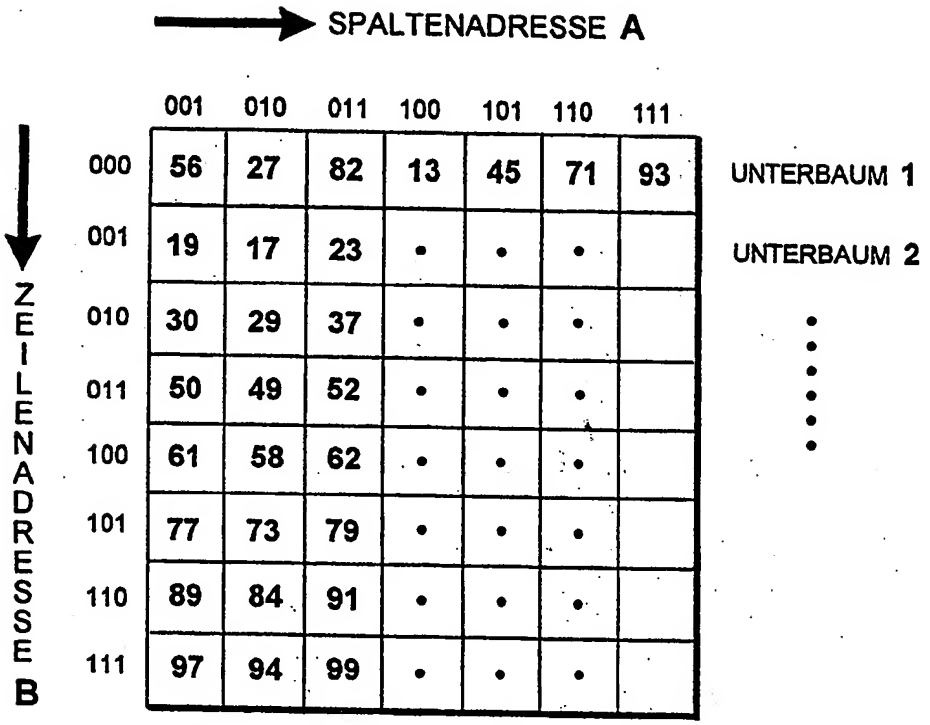



FIG.10

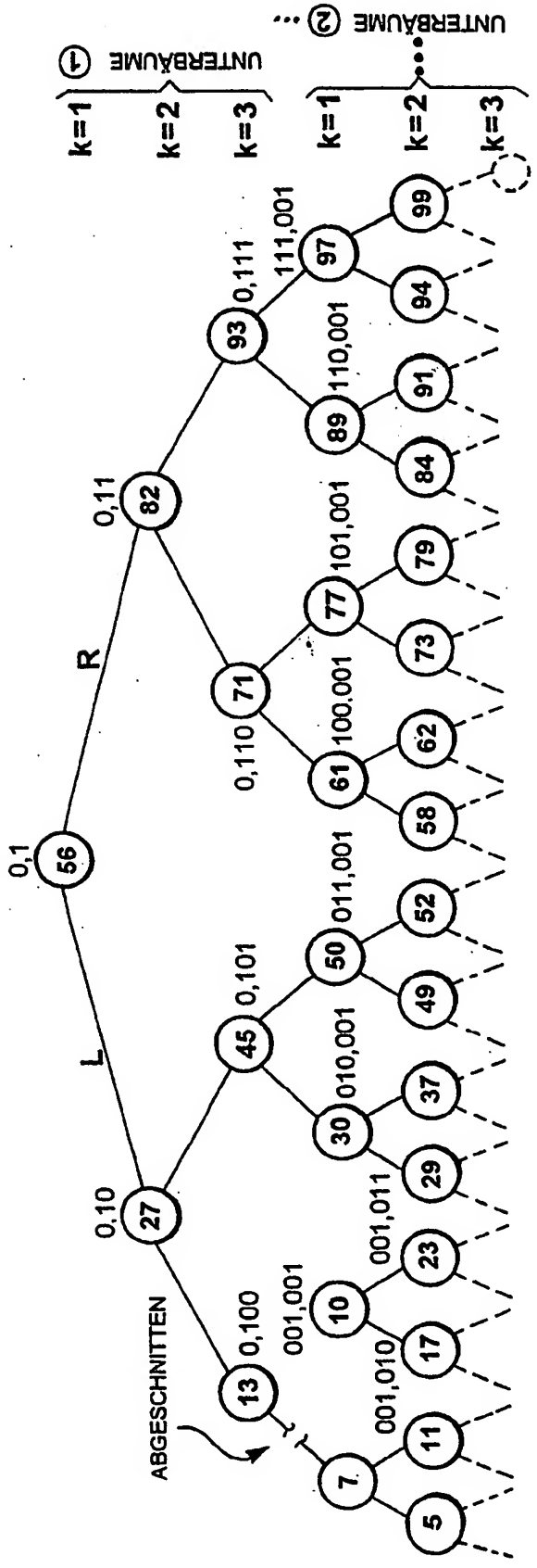
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

# FIG.11



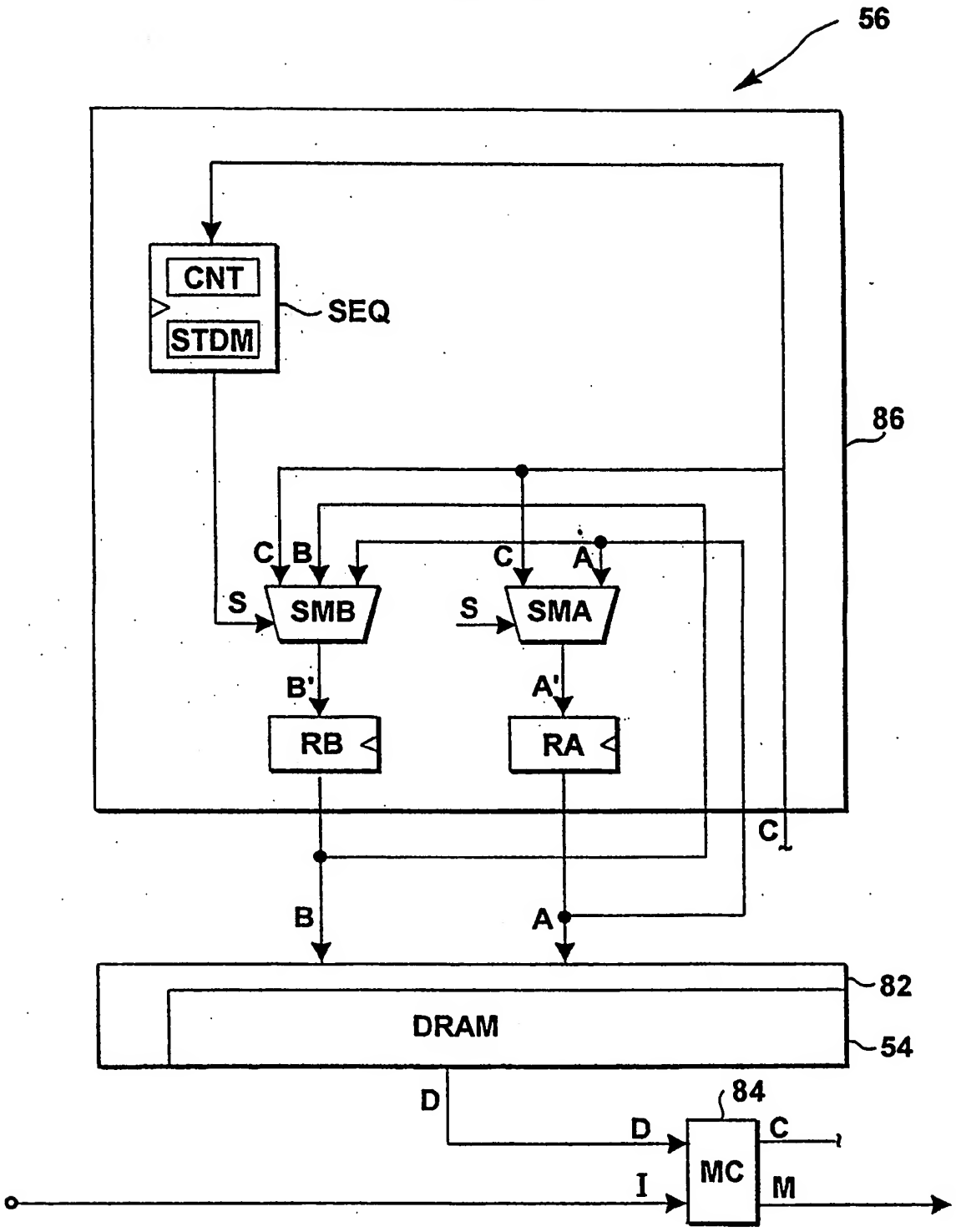
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.12



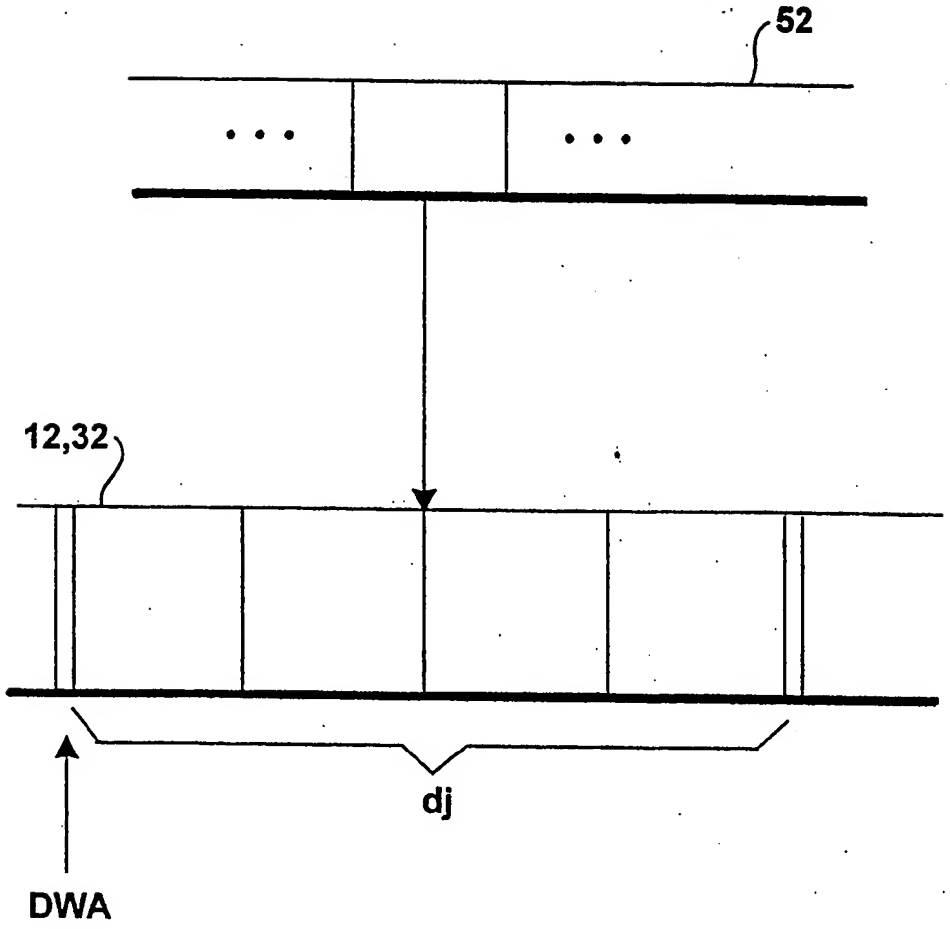
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.13



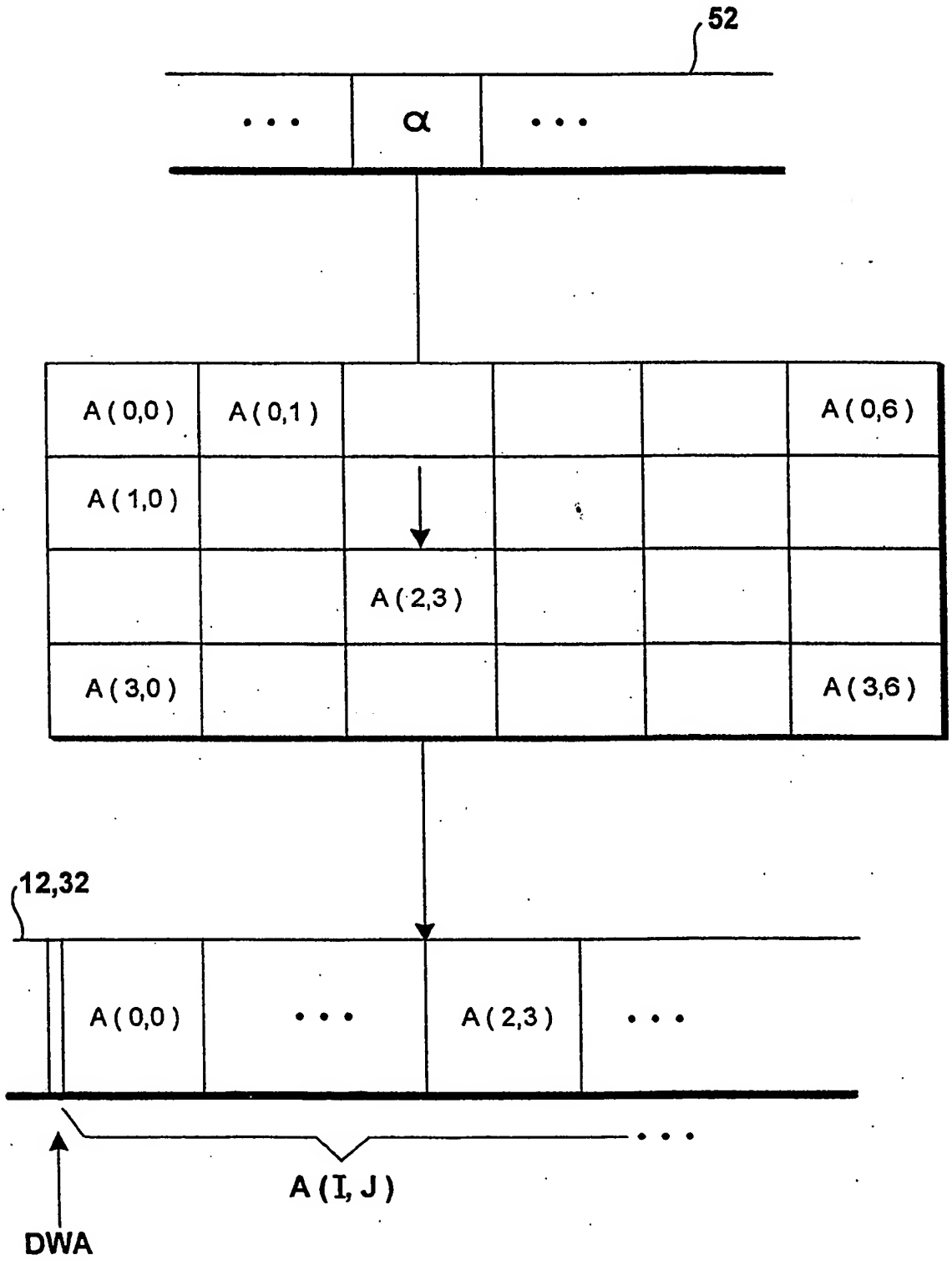
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.14



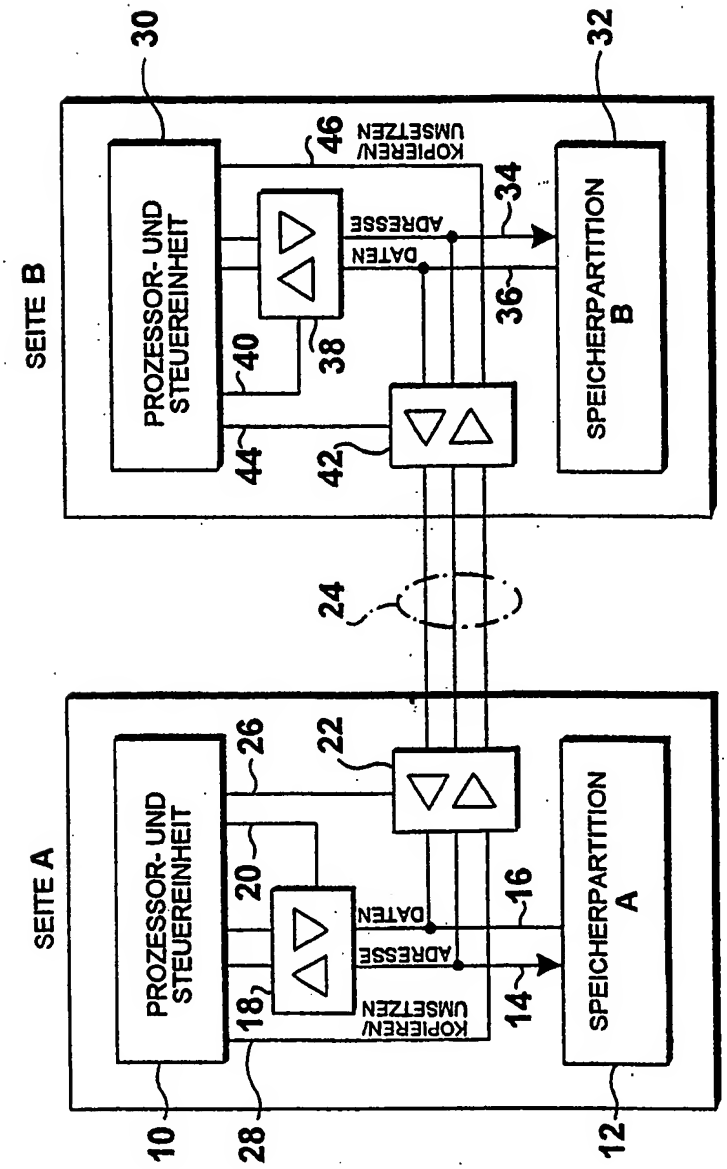
Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.15



Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.16



Nummer: DE 198 10 784 A1
Int. Cl.[6]: G 06 F 9/445
Offenlegungstag: 16. September 1999

FIG.17